Backward analysis

Lecture 27
December 4, 2014

Part I

Backward analysis

Backward analysis

1. $\mathsf{P} = \langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be a random ordering of n distinct numbers.
2. $X_i = 1 \iff \mathbf{p}_i$ is smaller than $\mathbf{p}_1, \ldots, \mathbf{p}_{i-1}$.
3. **Lemma**

   $\mathbf{Pr}[X_i = 1] = 1/i$.

Backward analysis

1. $\mathbf{P} = \langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be a random ordering of n distinct numbers.
2. $X_i = 1 \iff \mathbf{p}_i$ is smaller than $\mathbf{p}_1, \ldots, \mathbf{p}_{i-1}$.
3. **Lemma**

   $\Pr[X_i = 1] = 1/i$.

Backward analysis

1. $\mathbf{P} = \langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be a random ordering of n distinct numbers.
2. $X_i = 1 \quad \iff \quad \mathbf{p}_i$ is smaller than $\mathbf{p}_1, \ldots, \mathbf{p}_{i-1}$.
3. **Lemma**

   $\Pr[X_i = 1] = 1/i$.

Proof...

Lemma

$\Pr[X_i = 1] = 1/i$.

Proof.

1. Fix elements appearing in $\mathrm{set}(\mathsf{P}_i) = \{s_1, \ldots, s_i\}$.
2. $\Pr\left[\mathsf{p}_i = \min(\mathsf{P}_i) \;\middle|\; \mathrm{set}(\mathsf{P}_i)\right] = 1/i$.

Proof...

Lemma

$\Pr[X_i = 1] = 1/i$.

Proof.

1. Fix elements appearing in $\mathrm{set}(\mathsf{P}_i) = \{s_1, \ldots, s_i\}$.
2. $\Pr\Big[\mathsf{p}_i = \min(\mathsf{P}_i) \;\Big|\; \mathrm{set}(\mathsf{P}_i)\Big] = 1/i$.

Proof...

Lemma

$\Pr[X_i = 1] = 1/i.$

Proof.

1. Fix elements appearing in $\mathrm{set}(\mathsf{P}_i) = \{s_1, \ldots, s_i\}$.
2. $\Pr\Big[\mathbf{p}_i = \min(\mathsf{P}_i) \;\Big|\; \mathrm{set}(\mathsf{P}_i)\Big] = 1/i$.

$$\Pr\Big[\mathbf{p}_i = \min(\mathsf{P}_i)\Big]$$

Proof...

Lemma

$\Pr[X_i = 1] = 1/i$.

Proof.

1. Fix elements appearing in $\mathrm{set}(\mathsf{P}_i) = \{s_1, \ldots, s_i\}$.
2. $\Pr\Big[\mathsf{p}_i = \min(\mathsf{P}_i) \;\Big|\, \mathrm{set}(\mathsf{P}_i)\Big] = 1/i$.

$$\Pr\Big[\mathsf{p}_i = \min(\mathsf{P}_i)\Big] = \sum_{S \subseteq \mathsf{P}, |S| = i} \Pr\Big[\mathsf{p}_i = \min(\mathsf{P}_i) \;\Big|\, \mathrm{set}(\mathsf{P}_i) = S\Big] \Pr[S]$$

Proof...

Lemma

$\Pr[X_i = 1] = 1/i$.

Proof.

1. Fix elements appearing in $\mathrm{set}(\mathbf{P}_i) = \{s_1, \ldots, s_i\}$.
2. $\Pr\Big[\mathbf{p}_i = \min(\mathbf{P}_i) \;\Big|\; \mathrm{set}(\mathbf{P}_i)\Big] = 1/i$.

$$
\begin{aligned}
\Pr\Big[\mathbf{p}_i = \min(\mathbf{P}_i)\Big] &= \sum_{S \subseteq \mathbf{P}, |S| = i} \Pr\Big[\mathbf{p}_i = \min(\mathbf{P}_i) \;\Big|\; \mathrm{set}(\mathbf{P}_i) = S\Big] \Pr[S] \\
&= \sum_{S \subseteq \mathbf{P}, |S| = i} \frac{1}{i} \Pr[S]
\end{aligned}
$$

Proof...

Lemma

$\Pr[X_i = 1] = 1/i$.

Proof.

1. Fix elements appearing in $\mathrm{set}(\mathsf{P}_i) = \{s_1, \ldots, s_i\}$.
2. $\Pr\Big[\mathsf{p}_i = \min(\mathsf{P}_i) \;\Big|\, \mathrm{set}(\mathsf{P}_i)\Big] = 1/i$.

$$
\begin{aligned}
&\Pr\Big[\mathsf{p}_i = \min(\mathsf{P}_i)\Big] \\
&\quad = \sum_{S \subseteq \mathsf{P}, |S| = i} \Pr\Big[\mathsf{p}_i = \min(\mathsf{P}_i) \;\Big|\, \mathrm{set}(\mathsf{P}_i) = S\Big] \Pr[S] \\
&\quad = \sum_{S \subseteq \mathsf{P}, |S| = i} \frac{1}{i} \Pr[S] = \frac{1}{i}.
\end{aligned}
$$

Proof...

Lemma

$\Pr[X_i = 1] = 1/i$.

Proof.

1. Fix elements appearing in $\mathrm{set}(\mathbf{P}_i) = \{s_1, \ldots, s_i\}$.
2. $\Pr\Big[\mathbf{p}_i = \min(\mathbf{P}_i) \;\Big|\; \mathrm{set}(\mathbf{P}_i)\Big] = 1/i$.

$$\begin{aligned}
&\Pr\Big[\mathbf{p}_i = \min(\mathbf{P}_i)\Big] \\
&= \sum_{S \subseteq \mathbf{P}, |S| = i} \Pr\Big[\mathbf{p}_i = \min(\mathbf{P}_i) \;\Big|\; \mathrm{set}(\mathbf{P}_i) = S\Big] \Pr[S] \\
&= \sum_{S \subseteq \mathbf{P}, |S| = i} \frac{1}{i} \Pr[S] = \frac{1}{i}. \qquad \square
\end{aligned}$$

$\#$ of times...

...the minimum changes in a random permutation...

Theorem

In a random permutation of n distinct numbers, the minimum of the prefix changes in expectation $\ln n + 1$ times.

Proof.

1. $Y = \sum_{i=1}^{n} X_i$.
2. $\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \leq \ln n + 1$.

∎

of times...

...the minimum changes in a random permutation...

Theorem

In a random permutation of n distinct numbers, the minimum of the prefix changes in expectation $\ln n + 1$ times.

Proof.

1. $Y = \sum_{i=1}^{n} X_i$.
2. $\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \leq \ln n + 1.$

$\square$

of times...

...the minimum changes in a random permutation...

Theorem

In a random permutation of n distinct numbers, the minimum of the prefix changes in expectation $\ln n + 1$ times.

Proof.

1. $Y = \sum_{i=1}^{n} X_i$.
2. $\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \leq \ln n + 1.$

∎

$\#$ of times...

...the minimum changes in a random permutation...

Theorem

In a random permutation of n distinct numbers, the minimum of the prefix changes in expectation $\ln n + 1$ times.

Proof.

1. $Y = \sum_{i=1}^{n} X_i$.
2. $\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \leq \ln n + 1$.

$\square$

of times...

...the minimum changes in a random permutation...

Theorem

In a random permutation of n distinct numbers, the minimum of the prefix changes in expectation $\ln n + 1$ times.

Proof.

1. $Y = \sum_{i=1}^{n} X_i$.
2. $\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \leq \ln n + 1.$

□

of times...

...the minimum changes in a random permutation...

Theorem

In a random permutation of n *distinct numbers, the minimum of the prefix changes in expectation* $\ln n + 1$ *times.*

Proof.

1. $Y = \sum_{i=1}^{n} X_i$.
2. $\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \leq \ln n + 1$.

□

of times...

...the minimum changes in a random permutation...

Theorem

In a random permutation of n distinct numbers, the minimum of the prefix changes in expectation $\ln n + 1$ times.

Proof.

1. $Y = \sum_{i=1}^{n} X_i$.
2. $\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \leq \ln n + 1$.

$\square$

High probability

Lemma

$\Pi = \pi_1 \ldots \pi_n$: *random permutation of* $\{1, \ldots, n\}$. X_i: *indicator variable if* π_i *is the smallest number in* $\{\pi_1, \ldots, \pi_i\}$, *for* $\forall i$. *Then* $Z = \sum_{i=1}^{n} X_i = O(\log n)$*., w.h.p. (i.e.,* $\geq 1 - 1/n^{O(1)}$*).*

proof

1. $\mathcal{E}_i$: the event that $X_i = 1$, for $i = 1, \ldots, n$.
2. Claim: $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent.
3. Generate permutation: Randomly pick a permutation of the given numbers, set first number to be π_n.
4. Next, pick a random permutation of the remaining numbers and set the first number as π_{n-1} in output permutation.
5. Repeat this process till we generate the whole permutation.

High probability

Lemma

$\Pi = \pi_1 \ldots \pi_n$: *random permutation of* $\{1, \ldots, n\}$. X_i: *indicator variable if* π_i *is the smallest number in* $\{\pi_1, \ldots, \pi_i\}$, *for* $\forall i$. *Then* $Z = \sum_{i=1}^{n} X_i = O(\log n)$., *w.h.p. (i.e.,* $\geq 1 - 1/n^{O(1)}$*).*

proof

1. $\mathcal{E}_i$: the event that $X_i = 1$, for $i = 1, \ldots, n$.
2. Claim: $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent.
3. Generate permutation: Randomly pick a permutation of the given numbers, set first number to be π_n.
4. Next, pick a random permutation of the remaining numbers and set the first number as π_{n-1} in output permutation.
5. Repeat this process till we generate the whole permutation.

High probability

Lemma

$\Pi = \pi_1 \ldots \pi_n$: *random permutation of* $\{1, \ldots, n\}$. X_i: *indicator variable if* π_i *is the smallest number in* $\{\pi_1, \ldots, \pi_i\}$, *for* $\forall i$. *Then* $Z = \sum_{i=1}^{n} X_i = O(\log n)$., *w.h.p. (i.e.,* $\geq 1 - 1/n^{O(1)}$*).*

proof

1. $\mathcal{E}_i$: the event that $X_i = 1$, for $i = 1, \ldots, n$.
2. Claim: $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent.
3. Generate permutation: Randomly pick a permutation of the given numbers, set first number to be π_n.
4. Next, pick a random permutation of the remaining numbers and set the first number as π_{n-1} in output permutation.
5. Repeat this process till we generate the whole permutation.

High probability

Lemma

$\Pi = \pi_1 \ldots \pi_n$: *random permutation of* $\{1, \ldots, n\}$. X_i: *indicator variable if* π_i *is the smallest number in* $\{\pi_1, \ldots, \pi_i\}$, *for* $\forall i$. *Then* $Z = \sum_{i=1}^{n} X_i = O(\log n)$., *w.h.p. (i.e.,* $\geq 1 - 1/n^{O(1)}$*).*

proof

1. $\mathcal{E}_i$: the event that $X_i = 1$, for $i = 1, \ldots, n$.
2. Claim: $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent.
3. Generate permutation: Randomly pick a permutation of the given numbers, set first number to be π_n.
4. Next, pick a random permutation of the remaining numbers and set the first number as π_{n-1} in output permutation.
5. Repeat this process till we generate the whole permutation.

High probability

Lemma

$\Pi = \pi_1 \ldots \pi_n$: *random permutation of* $\{1, \ldots, n\}$. X_i: *indicator variable if* π_i *is the smallest number in* $\{\pi_1, \ldots, \pi_i\}$, *for* $\forall i$. *Then* $Z = \sum_{i=1}^{n} X_i = O(\log n)$., *w.h.p. (i.e.,* $\geq 1 - 1/n^{O(1)}$*).*

proof

1. $\mathcal{E}_i$: the event that $X_i = 1$, for $i = 1, \ldots, n$.
2. Claim: $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent.
3. Generate permutation: Randomly pick a permutation of the given numbers, set first number to be π_n.
4. Next, pick a random permutation of the remaining numbers and set the first number as π_{n-1} in output permutation.
5. Repeat this process till we generate the whole permutation.

Proof continued...

1. For any indices $1 \le i_1 < i_2 < \ldots < i_k \le n$, and observe that $\Pr\left[\mathcal{E}_{i_k} \,\middle|\, \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\right] = \Pr\left[\mathcal{E}_{i_k}\right]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\Pr\left[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] =$
   $\Pr\left[\mathcal{E}_{i_1} \,\middle|\, \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] \Pr[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] =$
   $\Pr\left[\mathcal{E}_{i_1}\right] \Pr\left[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\right] = \prod_{j=1}^{k} \Pr\left[\mathcal{E}_{i_j}\right] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ■

Proof continued...

1. For any indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, and observe that $\Pr\left[\mathcal{E}_{i_k} \,\middle|\, \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\right] = \Pr\left[\mathcal{E}_{i_k}\right]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\Pr\left[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] = \Pr\left[\mathcal{E}_{i_1} \,\middle|\, \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] \Pr[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] = \Pr\left[\mathcal{E}_{i_1}\right] \Pr\left[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\right] = \prod_{j=1}^{k} \Pr\left[\mathcal{E}_{i_j}\right] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ∎

Proof continued...

1. For any indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, and observe that $\Pr\left[\mathcal{E}_{i_k} \,\middle|\, \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\right] = \Pr\left[\mathcal{E}_{i_k}\right]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\Pr\left[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] =$ $\Pr\left[\mathcal{E}_{i_1} \,\middle|\, \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] \Pr[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] =$ $\Pr\left[\mathcal{E}_{i_1}\right] \Pr\left[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\right] = \prod_{j=1}^{k} \Pr\left[\mathcal{E}_{i_j}\right] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ■

Proof continued...

1. For any indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, and observe that $\mathbf{Pr}\Big[\mathcal{E}_{i_k} \;\Big|\; \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\Big] = \mathbf{Pr}\Big[\mathcal{E}_{i_k}\Big]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\mathbf{Pr}\Big[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\Big] =$ $\mathbf{Pr}\Big[\mathcal{E}_{i_1} \;\Big|\; \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\Big] \mathbf{Pr}[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] =$ $\mathbf{Pr}\Big[\mathcal{E}_{i_1}\Big] \mathbf{Pr}\Big[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\Big] = \prod_{j=1}^{k} \mathbf{Pr}\Big[\mathcal{E}_{i_j}\Big] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ∎

Proof continued...

1. For any indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, and observe that $\mathbf{Pr}\left[\mathcal{E}_{i_k} \;\middle|\; \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\right] = \mathbf{Pr}\left[\mathcal{E}_{i_k}\right]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\mathbf{Pr}\left[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] =$ $\mathbf{Pr}\left[\mathcal{E}_{i_1} \;\middle|\; \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] \mathbf{Pr}[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] =$ $\mathbf{Pr}\left[\mathcal{E}_{i_1}\right] \mathbf{Pr}\left[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\right] = \prod_{j=1}^{k} \mathbf{Pr}\left[\mathcal{E}_{i_j}\right] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ■

Proof continued...

1. For any indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, and observe that $\mathbf{Pr}\left[\mathcal{E}_{i_k} \,\middle|\, \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\right] = \mathbf{Pr}\left[\mathcal{E}_{i_k}\right]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\mathbf{Pr}\left[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] =$ $\mathbf{Pr}\left[\mathcal{E}_{i_1} \,\middle|\, \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] \mathbf{Pr}[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] =$ $\mathbf{Pr}\left[\mathcal{E}_{i_1}\right] \mathbf{Pr}\left[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\right] = \prod_{j=1}^{k} \mathbf{Pr}\left[\mathcal{E}_{i_j}\right] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ■

Proof continued...

1. For any indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, and observe that $\mathbf{Pr}\Big[\mathcal{E}_{i_k} \;\Big|\; \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\Big] = \mathbf{Pr}\Big[\mathcal{E}_{i_k}\Big]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\mathbf{Pr}\Big[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\Big] =$ $\mathbf{Pr}\Big[\mathcal{E}_{i_1} \;\Big|\; \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\Big] \mathbf{Pr}[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] =$ $\mathbf{Pr}\Big[\mathcal{E}_{i_1}\Big] \mathbf{Pr}\Big[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\Big] = \prod_{j=1}^{k} \mathbf{Pr}\Big[\mathcal{E}_{i_j}\Big] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ■

Proof continued...

1. For any indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, and observe that $\mathbf{Pr}\left[\mathcal{E}_{i_k} \,\middle|\, \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\right] = \mathbf{Pr}\left[\mathcal{E}_{i_k}\right]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\mathbf{Pr}\left[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] =$ $\mathbf{Pr}\left[\mathcal{E}_{i_1} \,\middle|\, \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] \mathbf{Pr}[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] =$ $\mathbf{Pr}\left[\mathcal{E}_{i_1}\right] \mathbf{Pr}\left[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\right] = \prod_{j=1}^{k} \mathbf{Pr}\left[\mathcal{E}_{i_j}\right] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ∎

Proof continued...

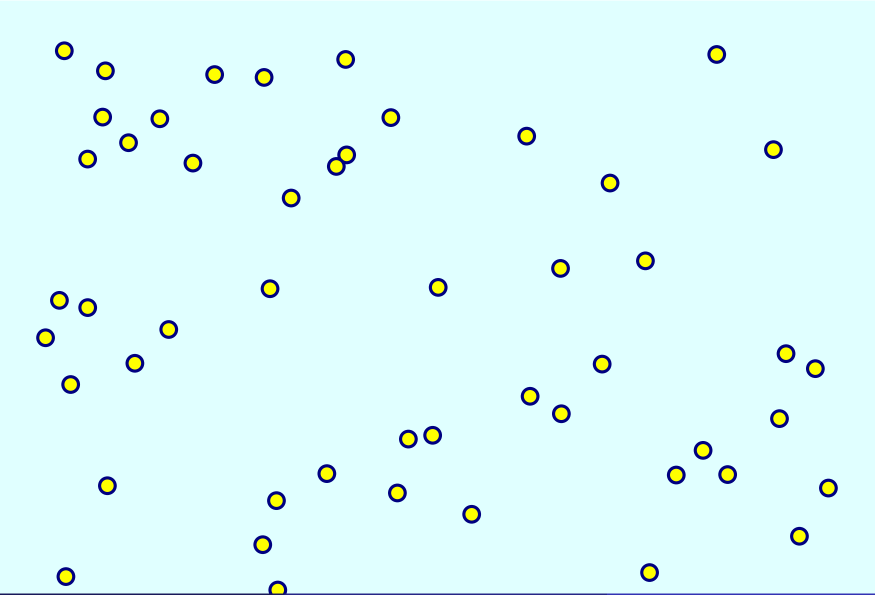
1. For any indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, and observe that $\mathbf{Pr}\left[\mathcal{E}_{i_k} \,\middle|\, \mathcal{E}_{i_1} \cap \ldots \cap \mathcal{E}_{i_{k-1}}\right] = \mathbf{Pr}\left[\mathcal{E}_{i_k}\right]$,
2. ..because $\mathcal{E}_{i_1}$ determined after all $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_k$.
3. By induction: $\mathbf{Pr}\left[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] =$ $\mathbf{Pr}\left[\mathcal{E}_{i_1} \,\middle|\, \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\right] \mathbf{Pr}[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}] =$ $\mathbf{Pr}\left[\mathcal{E}_{i_1}\right] \mathbf{Pr}\left[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_3} \cap \ldots \cap \mathcal{E}_{i_k}\right] = \prod_{j=1}^{k} \mathbf{Pr}\left[\mathcal{E}_{i_j}\right] = \prod_{j=1}^{k} \frac{1}{i_j}$.
4. $\implies$ variables $X_1, \ldots, X_n$ are independent.
5. Result readily follows from Chernoff's inequality. ■

Part II

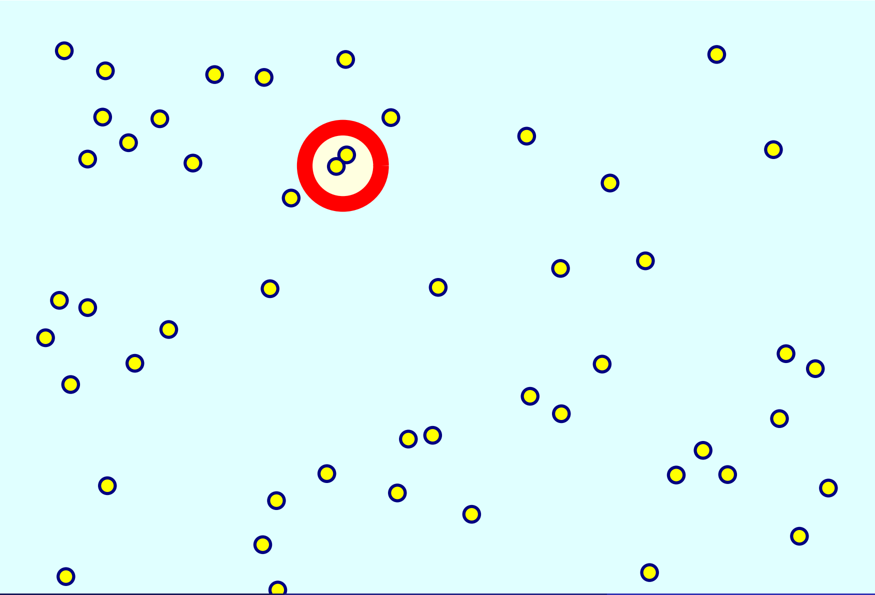
Closet pair in linear time

Finding the closest pair of points

Finding the closest pair of points

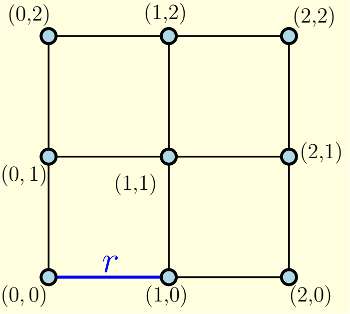


Finding the closest pair of points



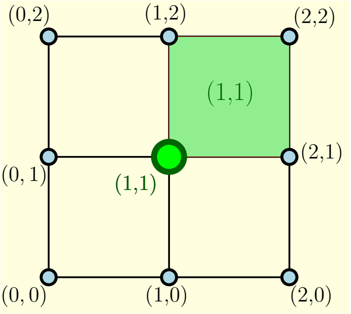
Grids...

1. r: Side length of grid cell.
2. Grid cell IDed by pair of integers.
3. Constant time to determine a point p's grid cell id:
$$\boxed{id(p) = (\lfloor p_x/r \rfloor, \lfloor p_y/r \rfloor)}$$
4. Limited use of the floor function (but no word packing tricks).
5. Use hashing on (grid) points.
6. Store points in grid...
...in linear time.



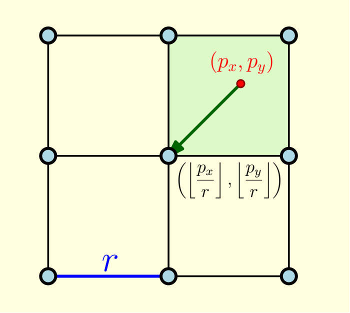
Grids...

1. r: Side length of grid cell.
2. Grid cell IDed by pair of integers.
3. Constant time to determine a point p's grid cell id: $\boxed{id(p) = (\lfloor p_x/r \rfloor, \lfloor p_y/r \rfloor)}$
4. Limited use of the floor function (but no word packing tricks).
5. Use hashing on (grid) points.
6. Store points in grid... ...in linear time.



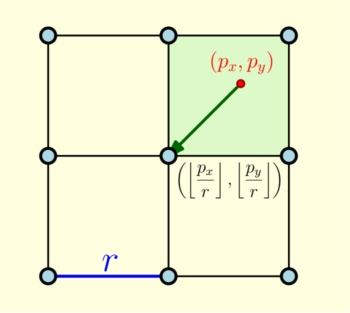
Grids...

1. r: Side length of grid cell.
2. Grid cell IDed by pair of integers.
3. Constant time to determine a point p's grid cell id:
   $\boxed{id(p) = (\lfloor p_x/r \rfloor, \lfloor p_y/r \rfloor)}$
4. Limited use of the floor function (but no word packing tricks).
5. Use hashing on (grid) points.
6. Store points in grid...
   ...in linear time.



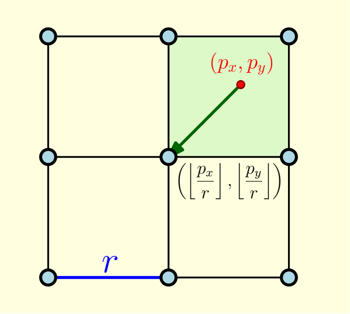
Grids...

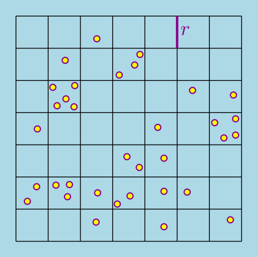
1. r: Side length of grid cell.
2. Grid cell IDed by pair of integers.
3. Constant time to determine a point p's grid cell id:
   $$\boxed{id(p) = (\lfloor p_x/r \rfloor, \lfloor p_y/r \rfloor)}$$
4. Limited use of the floor function (but no word packing tricks).
5. Use hashing on (grid) points.
6. Store points in grid...
   ...in linear time.



Grids...

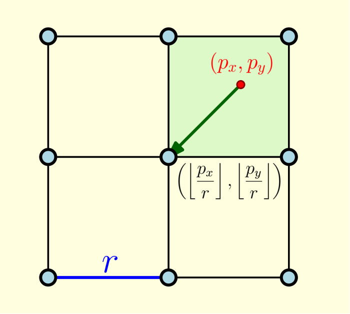
1. r: Side length of grid cell.
2. Grid cell IDed by pair of integers.
3. Constant time to determine a point p's grid cell id: $\boxed{id(p) = (\lfloor p_x/r \rfloor, \lfloor p_y/r \rfloor)}$
4. Limited use of the floor function (but no word packing tricks).
5. Use hashing on (grid) points.
6. Store points in grid... ...in linear time.

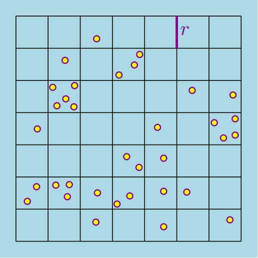




Grids...

1. r: Side length of grid cell.
2. Grid cell IDed by pair of integers.
3. Constant time to determine a point p's grid cell id:
   $$\boxed{id(p) = \left(\lfloor p_x/r \rfloor, \lfloor p_y/r \rfloor\right)}$$
4. Limited use of the floor function (but no word packing tricks).
5. Use hashing on (grid) points.
6. Store points in grid... ...in linear time.

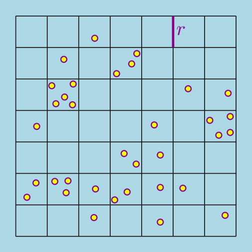




Storing point set in grid/hash-table...

Hashing:

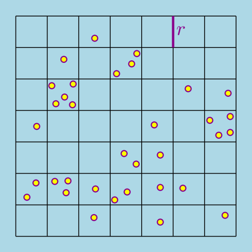
1. Non-empty grid cells
2. For non-empty grid cell:
   List of points in it.
3. For a grid cell:
   Its neighboring cells.



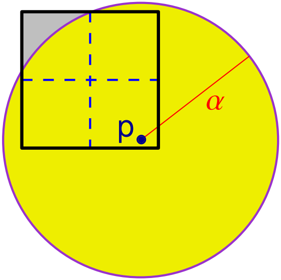
Storing point set in grid/hash-table...

Hashing:

1. Non-empty grid cells
2. For non-empty grid cell:
   List of points in it.
3. For a grid cell:
   Its neighboring cells.



Closet pair in a square



Lemma

Let P *be a set of points contained inside a square* $\Box$*, such that the sidelength of* $\Box$ *is* $\alpha = \mathcal{CP}(\mathsf{P})$*. Then* $|\mathsf{P}| \leq 4$*.*

Proof.

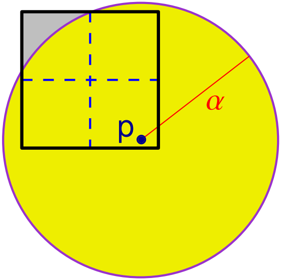
Partition $\Box$ into four equal squares $\Box_1, \ldots, \Box_4$.

Each square diameter $\sqrt{2}\alpha/2 < \alpha$.

... contain at most one point of P; that is, the disk of radius α centered at a point $\mathsf{p} \in \mathsf{P}$ completely covers the subsquare containing it; see the figure on the right.

P can have four points if it is the four corners of $\Box$. $\Box$

Closet pair in a square



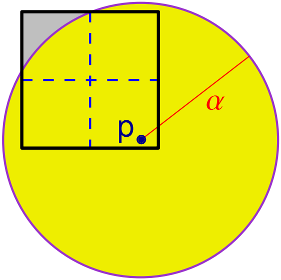
Lemma

Let $\mathbf{P}$ be a set of points contained inside a square $\square$, such that the sidelength of $\square$ is $\alpha = \mathcal{CP}(\mathbf{P})$. Then $|\mathbf{P}| \leq 4$.

Proof.

Partition $\square$ into four equal squares $\square_1, \ldots, \square_4$.
Each square diameter $\sqrt{2}\alpha/2 < \alpha$.
... contain at most one point of $\mathbf{P}$; that is, the disk of radius α centered at a point $\mathbf{p} \in \mathbf{P}$ completely covers the subsquare containing it; see the figure on the right.
$\mathbf{P}$ can have four points if it is the four corners of $\square$. $\square$

Closet pair in a square



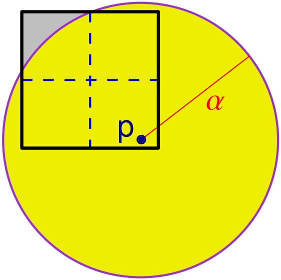
Lemma

Let $\mathbf{P}$ *be a set of points contained inside a square* $\square$*, such that the sidelength of* $\square$ *is* $\alpha = \mathcal{CP}(\mathbf{P})$*. Then* $|\mathbf{P}| \leq 4$*.*

Proof.

Partition $\square$ into four equal squares $\square_1, \ldots, \square_4$.
Each square diameter $\sqrt{2}\alpha/2 < \alpha$.
... contain at most one point of $\mathbf{P}$; that is, the disk of radius α centered at a point $\mathbf{p} \in \mathbf{P}$ completely covers the subsquare containing it; see the figure on the right.
$\mathbf{P}$ can have four points if it is the four corners of $\square$. $\square$

Closet pair in a square



Lemma

Let P *be a set of points contained inside a square* $\square$*, such that the sidelength of* $\square$ *is* $\alpha = \mathcal{CP}(\mathsf{P})$*. Then* $|\mathsf{P}| \leq 4$*.*

Proof.

Partition $\square$ into four equal squares $\square_1, \ldots, \square_4$.
Each square diameter $\sqrt{2}\alpha/2 < \alpha$.
... contain at most one point of P; that is, the disk of radius α centered at a point $\mathsf{p} \in \mathsf{P}$ completely covers the subsquare containing it; see the figure on the right.
P can have four points if it is the four corners of $\square$. $\square$

Verify closet pair

Lemma

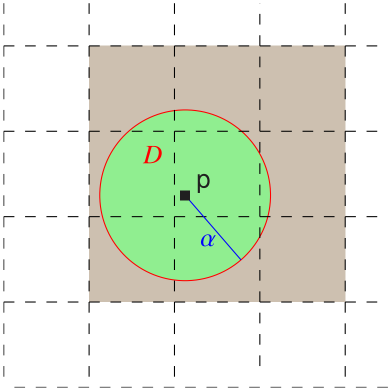
P: *set of* n *points in the plane.* α: *distance. Verify in linear time whether* $\mathcal{CP}(\mathsf{P}) < \alpha$, $\mathcal{CP}(\mathsf{P}) = \alpha$, *or* $\mathcal{CP}(\mathsf{P}) > \alpha$.

proof

Indeed, store the points of **P** in the grid G_α. For every non-empty grid cell, we maintain a linked list of the points inside it. Thus, adding a new point **p** takes constant time. Specifically, compute $\mathrm{id}(\mathbf{p})$, check if $\mathrm{id}(\mathbf{p})$ already appears in the hash table, if not, create a new linked list for the cell with this ID number, and store **p** in it. If a linked list already exists for $\mathrm{id}(\mathbf{p})$, just add **p** to it. This takes $O(n)$ time overall.

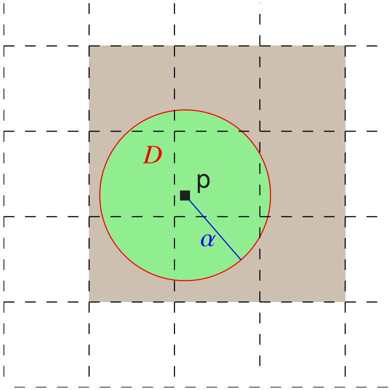
Now, if any grid cell in $\mathsf{G}_\alpha(\mathsf{P})$ contains more than, say, 4 points of **P**, then it must be that the $\mathcal{CP}(\mathsf{P}) < \alpha$, by previous lemma.

Proof continued



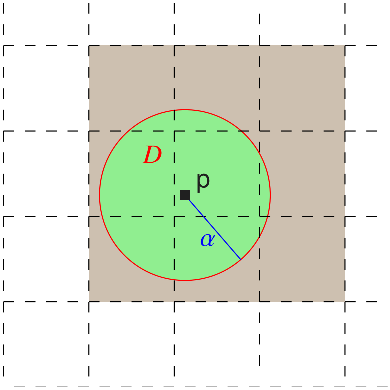
1. When insert a point **p**: fetch all the points of **P** in cluster of **P**
2. Takes constant time.
3. If there is a point closer to **p** than α that was already inserted, then it must be stored in one of these 9 cells.
4. Now, each one of those cells must contain at most 4 points of **P** by prev lemma.
5. Otherwise, already stopped since $\mathcal{CP}(\cdot) < \alpha$.

Proof continued



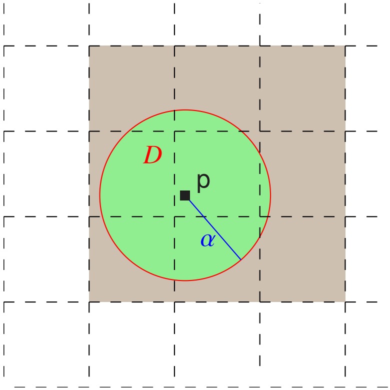
1. When insert a point **p**: fetch all the points of **P** in cluster of **P**
2. Takes constant time.
3. If there is a point closer to **p** than α that was already inserted, then it must be stored in one of these 9 cells.
4. Now, each one of those cells must contain at most 4 points of **P** by prev lemma.
5. Otherwise, already stopped since $\mathcal{CP}(\cdot) < \alpha$.

Proof continued



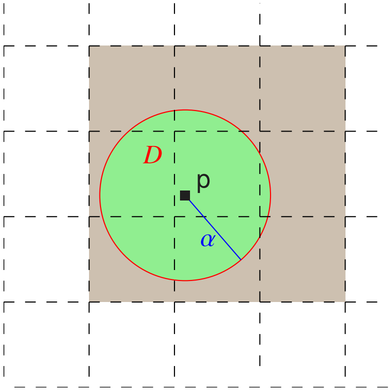
1. When insert a point $\mathbf{p}$: fetch all the points of $\mathbf{P}$ in cluster of $\mathbf{P}$
2. Takes constant time.
3. If there is a point closer to $\mathbf{p}$ than α that was already inserted, then it must be stored in one of these 9 cells.
4. Now, each one of those cells must contain at most 4 points of $\mathbf{P}$ by prev lemma.
5. Otherwise, already stopped since $\mathcal{CP}(\cdot) < \alpha$.

Proof continued



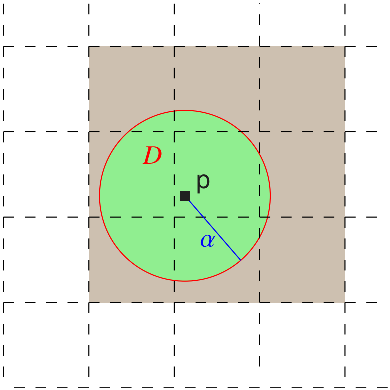
1. When insert a point **p**: fetch all the points of **P** in cluster of **P**
2. Takes constant time.
3. If there is a point closer to **p** than α that was already inserted, then it must be stored in one of these 9 cells.
4. Now, each one of those cells must contain at most 4 points of **P** by prev lemma.
5. Otherwise, already stopped since $\mathcal{CP}(\cdot) < \alpha$.

Proof continued



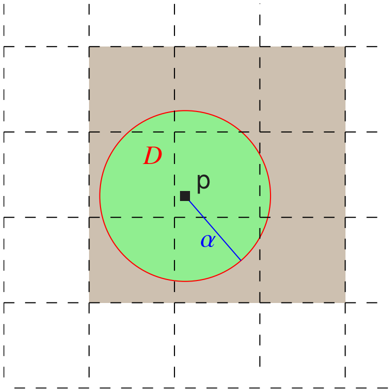
1. When insert a point **p**: fetch all the points of **P** in cluster of **P**
2. Takes constant time.
3. If there is a point closer to **p** than α that was already inserted, then it must be stored in one of these 9 cells.
4. Now, each one of those cells must contain at most 4 points of **P** by prev lemma.
5. Otherwise, already stopped since $\mathcal{CP}(\cdot) < \alpha$.

Proof continued



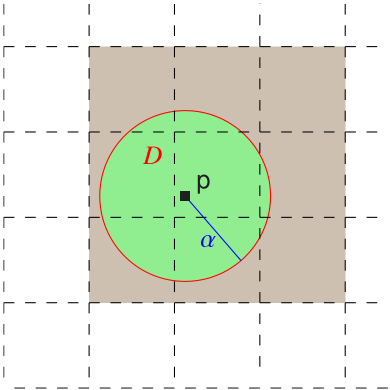
1. S set of all points in cluster.
2. $|S| \leq 9 \cdot 4 = O(1)$.
3. Compute closest point to $\mathbf{p}$ in S. $O(1)$ time.
4. If $\mathrm{d}(\mathbf{p}, S) < \alpha$, we stop; otherwise, continue to next point.
5. Correctness: '$\mathcal{CP}(\mathbf{P}) < \alpha$' returned only if such pair found.

Proof continued



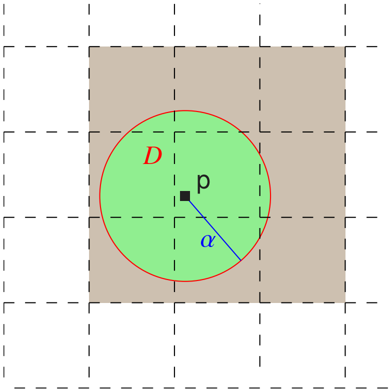
1. S set of all points in cluster.
2. $|S| \leq 9 \cdot 4 = O(1)$.
3. Compute closest point to $\mathbf{p}$ in S. $O(1)$ time.
4. If $\mathrm{d}(\mathbf{p}, S) < \alpha$, we stop; otherwise, continue to next point.
5. Correctness: '$\mathcal{CP}(\mathbf{P}) < \alpha$' returned only if such pair found.

Proof continued



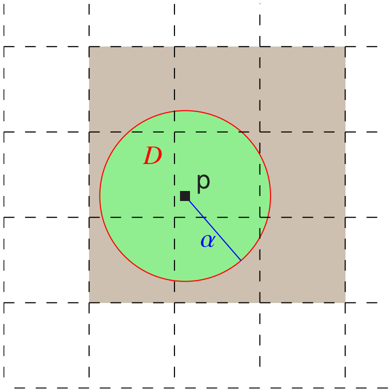
1. S set of all points in cluster.
2. $|S| \leq 9 \cdot 4 = O(1)$.
3. Compute closest point to $\mathbf{p}$ in S. $O(1)$ time.
4. If $\mathrm{d}(\mathbf{p}, S) < \alpha$, we stop; otherwise, continue to next point.
5. Correctness: '$\mathcal{CP}(\mathbf{P}) < \alpha$' returned only if such pair found.

Proof continued



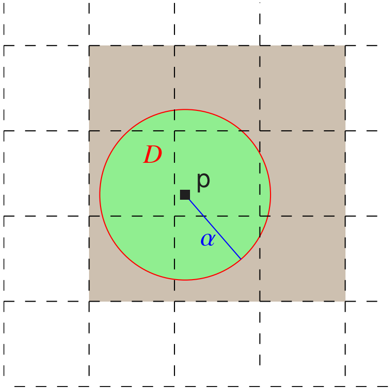
1. S set of all points in cluster.
2. $|S| \leq 9 \cdot 4 = O(1)$.
3. Compute closest point to $\mathbf{p}$ in S. $O(1)$ time.
4. If $\mathrm{d}(\mathbf{p}, S) < \alpha$, we stop; otherwise, continue to next point.
5. Correctness: '$\mathcal{CP}(\mathbf{P}) < \alpha$' returned only if such pair found.

Proof continued



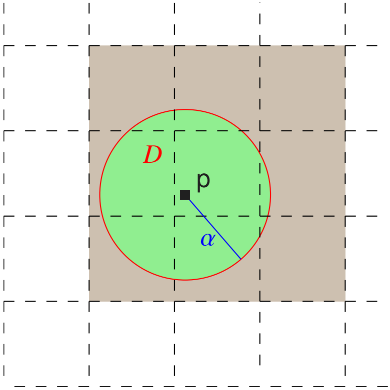
1. S set of all points in cluster.
2. $|S| \leq 9 \cdot 4 = O(1)$.
3. Compute closest point to **p** in S. $O(1)$ time.
4. If $\mathrm{d}(\mathbf{p}, S) < \alpha$, we stop; otherwise, continue to next point.
5. Correctness: '$\mathcal{CP}(\mathsf{P}) < \alpha$' returned only if such pair found.

Proof continued



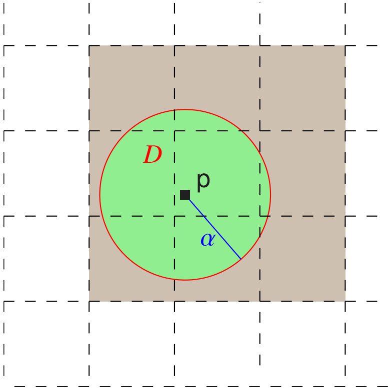
1. Assume **p** and **q**: realizing closest pair.
2. $\|\mathbf{p} - \mathbf{q}\| = \mathcal{CP}(\mathsf{P}) < \alpha$.
3. When later point (say **p**) inserted, the set S would contain **q**.
4. algorithm would stop and return '$\mathcal{CP}(\mathsf{P}) < \alpha$'.
5.

Proof continued



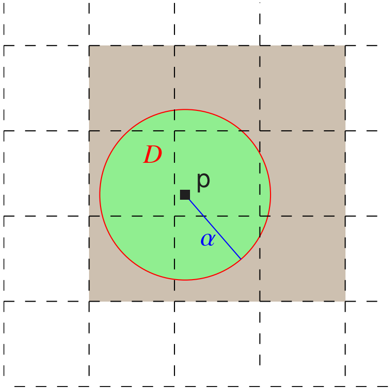
1. Assume **p** and **q**: realizing closest pair.
2. $\|\mathbf{p} - \mathbf{q}\| = \mathcal{CP}(\mathsf{P}) < \alpha$.
3. When later point (say **p**) inserted, the set S would contain **q**.
4. algorithm would stop and return '$\mathcal{CP}(\mathsf{P}) < \alpha$'.
5.

Proof continued



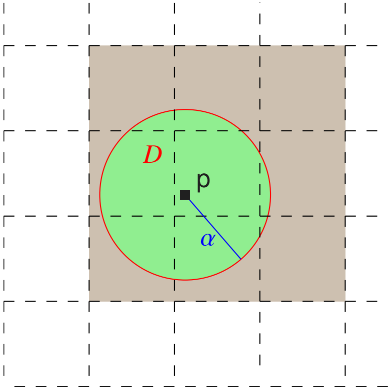
1. Assume **p** and **q**: realizing closest pair.
2. $\|\mathbf{p} - \mathbf{q}\| = \mathcal{CP}(\mathsf{P}) < \alpha$.
3. When later point (say **p**) inserted, the set S would contain **q**.
4. algorithm would stop and return '$\mathcal{CP}(\mathsf{P}) < \alpha$'.
5.

Proof continued



1. Assume **p** and **q**: realizing closest pair.
2. $\|\mathbf{p} - \mathbf{q}\| = \mathcal{CP}(\mathbf{P}) < \alpha$.
3. When later point (say **p**) inserted, the set S would contain **q**.
4. algorithm would stop and return '$\mathcal{CP}(\mathbf{P}) < \alpha$'.
5.

Proof continued



1. Assume **p** and **q**: realizing closest pair.
2. $\|\mathbf{p} - \mathbf{q}\| = \mathcal{CP}(\mathsf{P}) < \alpha$.
3. When later point (say **p**) inserted, the set S would contain **q**.
4. algorithm would stop and return '$\mathcal{CP}(\mathsf{P}) < \alpha$'.
5. ∎

New algorithm

1. Pick a random permutation of the points of P.
2. $\langle \mathsf{p}_1, \ldots, \mathsf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathsf{p}_1 - \mathsf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathsf{P}_i = \{\mathsf{p}_1, \ldots, \mathsf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
   1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
   2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
   3. rebuild the grid, and reinsert the i points of P_i from scratch into the grid G_{α_i}. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

New algorithm

1. Pick a random permutation of the points of P.
2. $\langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathsf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
    1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
    2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
    3. rebuild the grid, and reinsert the i points of P_i from scratch into the grid G_{α_i}. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

New algorithm

1. Pick a random permutation of the points of $\mathbf{P}$.
2. $\langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathbf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
    1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
    2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
    3. rebuild the grid, and reinsert the i points of $\mathbf{P}_i$ from scratch into the grid $\mathbf{G}_{\alpha_i}$. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

New algorithm

1. Pick a random permutation of the points of P.
2. $\langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathsf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
   1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
   2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
   3. rebuild the grid, and reinsert the i points of P_i from scratch into the grid G_{α_i}. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

New algorithm

1. Pick a random permutation of the points of $\mathbf{P}$.
2. $\langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathbf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
    1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
    2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
    3. rebuild the grid, and reinsert the i points of $\mathbf{P}_i$ from scratch into the grid $\mathbf{G}_{\alpha_i}$. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

New algorithm

1. Pick a random permutation of the points of P.
2. $\langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathsf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
    1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
    2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
    3. rebuild the grid, and reinsert the i points of P_i from scratch into the grid G_{α_i}. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

New algorithm

1. Pick a random permutation of the points of P.
2. $\langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathsf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
    1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
    2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
    3. rebuild the grid, and reinsert the i points of P_i from scratch into the grid G_{α_i}. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

New algorithm

1. Pick a random permutation of the points of P.
2. $\langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathsf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
   1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
   2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
   3. rebuild the grid, and reinsert the i points of P_i from scratch into the grid G_{α_i}. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

New algorithm

1. Pick a random permutation of the points of P.
2. $\langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$ be this permutation.
3. $\alpha_2 = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
4. Insert points into the closet-pair distance verifying data-structure.
5. α_i: the closest pair distance in the set $\mathsf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, for $i = 2, \ldots, n$.
6. ith iteration:
    1. if $\alpha_i = \alpha_{i-1}$. insertion takes constant time.
    2. If $\alpha_i < \alpha_{i-1}$ then: know new closest pair distance α_i.
    3. rebuild the grid, and reinsert the i points of P_i from scratch into the grid G_{α_i}. Takes $O(i)$ time.
7. Returns the number α_n and points realizing it.

Weak analysis...

Lemma

Let t be the number of different values in the sequence $\alpha_2, \alpha_3, \ldots, \alpha_n$. Then $\mathbf{E}[t] = O(\log n)$. As such, in expectation, the above algorithm rebuilds the grid $O(\log n)$ times.

proof

1. $X_i = 1 \iff \alpha_i < \alpha_{i-1}$.
2. $\mathbf{E}[X_i] = \Pr[X_i = 1]$ and $t = \sum_{i=3}^{n} X_i$.
3. $\Pr[X_i = 1] = \Pr[\alpha_i < \alpha_{i-1}]$.
4. Backward analysis. Fix P_i.
5. $\mathsf{q} \in \mathsf{P}_i$ is ***critical*** if $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{q}\}) > \mathcal{CP}(\mathsf{P}_i)$.
6. No critical points, then $\alpha_{i-1} = \alpha_i$ and then $\Pr[X_i = 1] = 0$.

Weak analysis...

Lemma

Let t be the number of different values in the sequence $\alpha_2, \alpha_3, \ldots, \alpha_n$. Then $\mathbf{E}[t] = O(\log n)$. As such, in expectation, the above algorithm rebuilds the grid $O(\log n)$ times.

proof

1. $X_i = 1 \iff \alpha_i < \alpha_{i-1}$.
2. $\mathbf{E}[X_i] = \Pr[X_i = 1]$ and $t = \sum_{i=3}^{n} X_i$.
3. $\Pr[X_i = 1] = \Pr[\alpha_i < \alpha_{i-1}]$.
4. Backward analysis. Fix P_i.
5. $\mathsf{q} \in \mathsf{P}_i$ is ***critical*** if $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{q}\}) > \mathcal{CP}(\mathsf{P}_i)$.
6. No critical points, then $\alpha_{i-1} = \alpha_i$ and then $\Pr[X_i = 1] = 0$.

Weak analysis...

Lemma

Let t be the number of different values in the sequence $\alpha_2, \alpha_3, \ldots, \alpha_n$. Then $\mathbf{E}[t] = O(\log n)$. As such, in expectation, the above algorithm rebuilds the grid $O(\log n)$ times.

proof

1. $X_i = 1 \iff \alpha_i < \alpha_{i-1}$.
2. $\mathbf{E}[X_i] = \Pr[X_i = 1]$ and $t = \sum_{i=3}^{n} X_i$.
3. $\Pr[X_i = 1] = \Pr[\alpha_i < \alpha_{i-1}]$.
4. Backward analysis. Fix P_i.
5. $\mathsf{q} \in \mathsf{P}_i$ is ***critical*** if $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{q}\}) > \mathcal{CP}(\mathsf{P}_i)$.
6. No critical points, then $\alpha_{i-1} = \alpha_i$ and then $\Pr[X_i = 1] = 0$.

Weak analysis...

Lemma

Let t be the number of different values in the sequence $\alpha_2, \alpha_3, \ldots, \alpha_n$. Then $\mathbf{E}[t] = O(\log n)$. As such, in expectation, the above algorithm rebuilds the grid $O(\log n)$ times.

proof

1. $X_i = 1 \iff \alpha_i < \alpha_{i-1}$.
2. $\mathbf{E}[X_i] = \Pr[X_i = 1]$ and $t = \sum_{i=3}^{n} X_i$.
3. $\Pr[X_i = 1] = \Pr[\alpha_i < \alpha_{i-1}]$.
4. Backward analysis. Fix P_i.
5. $\mathsf{q} \in \mathsf{P}_i$ is ***critical*** if $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{q}\}) > \mathcal{CP}(\mathsf{P}_i)$.
6. No critical points, then $\alpha_{i-1} = \alpha_i$ and then $\Pr[X_i = 1] = 0$.

Weak analysis...

Lemma

Let t be the number of different values in the sequence $\alpha_2, \alpha_3, \ldots, \alpha_n$. Then $\mathrm{E}[t] = O(\log n)$. As such, in expectation, the above algorithm rebuilds the grid $O(\log n)$ times.

proof

1. $X_i = 1 \iff \alpha_i < \alpha_{i-1}$.
2. $\mathrm{E}[X_i] = \Pr[X_i = 1]$ and $t = \sum_{i=3}^{n} X_i$.
3. $\Pr[X_i = 1] = \Pr[\alpha_i < \alpha_{i-1}]$.
4. Backward analysis. Fix P_i.
5. $\mathbf{q} \in \mathsf{P}_i$ is ***critical*** if $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathbf{q}\}) > \mathcal{CP}(\mathsf{P}_i)$.
6. No critical points, then $\alpha_{i-1} = \alpha_i$ and then $\Pr[X_i = 1] = 0$.

Weak analysis...

Lemma

Let t be the number of different values in the sequence $\alpha_2, \alpha_3, \ldots, \alpha_n$. Then $\mathbf{E}[t] = O(\log n)$. As such, in expectation, the above algorithm rebuilds the grid $O(\log n)$ times.

proof

1. $X_i = 1 \iff \alpha_i < \alpha_{i-1}$.
2. $\mathbf{E}[X_i] = \Pr[X_i = 1]$ and $t = \sum_{i=3}^{n} X_i$.
3. $\Pr[X_i = 1] = \Pr[\alpha_i < \alpha_{i-1}]$.
4. Backward analysis. Fix P_i.
5. $\mathsf{q} \in \mathsf{P}_i$ is ***critical*** if $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{q}\}) > \mathcal{CP}(\mathsf{P}_i)$.
6. No critical points, then $\alpha_{i-1} = \alpha_i$ and then $\Pr[X_i = 1] = 0$.

Proof continued...

1. If one critical point, then $\Pr[X_i = 1] = 1/i$.
2. Assume two critical points and let $\mathbf{p}, \mathbf{q}$ be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$.
3. $\alpha_i < \alpha_{i-1} \iff$ $\mathbf{p}$ or $\mathbf{q}$ is $\mathbf{p}_i$.
4. $\Pr[X_i = 1] = 2/i$.
5. Cannot be more than two critical points.
6. Linearity of expectations: $\mathbf{E}[t] = \mathbf{E}\left[\sum_{i=3}^{n} X_i\right] = \sum_{i=3}^{n} \mathbf{E}[X_i] \leq \sum_{i=3}^{n} 2/i = O(\log n)$.
7.

Proof continued...

1. If one critical point, then $\Pr[X_i = 1] = 1/i$.
2. Assume two critical points and let p, q be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$.
3. $\alpha_i < \alpha_{i-1} \iff$ p or q is p_i.
4. $\Pr[X_i = 1] = 2/i$.
5. Cannot be more than two critical points.
6. Linearity of expectations: $\mathbf{E}[t] = \mathbf{E}\left[\sum_{i=3}^{n} X_i\right] = \sum_{i=3}^{n} \mathbf{E}[X_i] \leq \sum_{i=3}^{n} 2/i = O(\log n)$.
7.

Proof continued...

1. If one critical point, then $\Pr[X_i = 1] = 1/i$.
2. Assume two critical points and let p, q be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$.
3. $\alpha_i < \alpha_{i-1} \iff$ p or q is p_i.
4. $\Pr[X_i = 1] = 2/i$.
5. Cannot be more than two critical points.
6. Linearity of expectations: $\mathbf{E}[t] = \mathbf{E}\left[\sum_{i=3}^{n} X_i\right] = \sum_{i=3}^{n} \mathbf{E}[X_i] \leq \sum_{i=3}^{n} 2/i = O(\log n)$.
7.

Proof continued...

1. If one critical point, then $\Pr[X_i = 1] = 1/i$.
2. Assume two critical points and let $\mathbf{p}, \mathbf{q}$ be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$.
3. $\alpha_i < \alpha_{i-1} \iff$ $\mathbf{p}$ or $\mathbf{q}$ is $\mathbf{p}_i$.
4. $\Pr[X_i = 1] = 2/i$.
5. Cannot be more than two critical points.
6. Linearity of expectations: $\mathbf{E}[t] = \mathbf{E}\left[\sum_{i=3}^{n} X_i\right] = \sum_{i=3}^{n} \mathbf{E}[X_i] \leq \sum_{i=3}^{n} 2/i = O(\log n)$.
7.

Proof continued...

1. If one critical point, then $\Pr[X_i = 1] = 1/i$.
2. Assume two critical points and let $\mathbf{p}, \mathbf{q}$ be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$.
3. $\alpha_i < \alpha_{i-1} \iff \mathbf{p}$ or $\mathbf{q}$ is $\mathbf{p}_i$.
4. $\Pr[X_i = 1] = 2/i$.
5. Cannot be more than two critical points.
6. Linearity of expectations: $\mathbf{E}[t] = \mathbf{E}\left[\sum_{i=3}^{n} X_i\right] = \sum_{i=3}^{n} \mathbf{E}[X_i] \leq \sum_{i=3}^{n} 2/i = O(\log n)$.
7.

Proof continued...

1. If one critical point, then $\Pr[X_i = 1] = 1/i$.
2. Assume two critical points and let $\mathbf{p}, \mathbf{q}$ be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$.
3. $\alpha_i < \alpha_{i-1} \iff \mathbf{p}$ or $\mathbf{q}$ is $\mathbf{p}_i$.
4. $\Pr[X_i = 1] = 2/i$.
5. Cannot be more than two critical points.
6. Linearity of expectations: $\mathbf{E}[t] = \mathbf{E}\left[\sum_{i=3}^{n} X_i\right] = \sum_{i=3}^{n} \mathbf{E}[X_i] \leq \sum_{i=3}^{n} 2/i = O(\log n)$.
7.

Proof continued...

1. If one critical point, then $\Pr[X_i = 1] = 1/i$.
2. Assume two critical points and let **p**, **q** be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$.
3. $\alpha_i < \alpha_{i-1} \iff$ **p** or **q** is $\mathbf{p}_i$.
4. $\Pr[X_i = 1] = 2/i$.
5. Cannot be more than two critical points.
6. Linearity of expectations: $\mathbf{E}[t] = \mathbf{E}\left[\sum_{i=3}^{n} X_i\right] = \sum_{i=3}^{n} \mathbf{E}[X_i] \leq \sum_{i=3}^{n} 2/i = O(\log n)$.
7. $\blacksquare$

Expected linear time analysis...

Theorem

P: *set of n points in the plane. Compute the closest pair of* **P** *in expected linear time.*

Proof.

1. $X_i = 1 \iff \alpha_i \neq \alpha_{i-1}$.
2. Running time is proportional to $R = 1 + \sum_{i=3}^{n}(1 + X_i \cdot i)$.
3. $\mathrm{E}[R] = \mathrm{E}\left[1 + \sum_{i=3}^{n}(1 + X_i \cdot i)\right] \leq n + \sum_{i=3}^{n} \mathrm{E}[X_i] \cdot i \leq n + \sum_{i=3}^{n} i \cdot \Pr[X_i = 1] \leq n + \sum_{i=3}^{n} i \cdot \frac{2}{i} \leq 3n$, by linearity of expectation and since $\mathrm{E}[X_i] = \Pr[X_i = 1] \leq 2/i$.
4. Expected running time of the algorithm is $O(\mathrm{E}[R]) = O(n)$. ∎

□

Expected linear time analysis...

Theorem

P: *set of* n *points in the plane. Compute the closest pair of* **P** *in expected linear time.*

Proof.

1. $X_i = 1 \iff \alpha_i \neq \alpha_{i-1}$.
2. Running time is proportional to $R = 1 + \sum_{i=3}^{n} (1 + X_i \cdot i)$.
3. $\mathrm{E}[R] = \mathrm{E}\Big[1 + \sum_{i=3}^{n} (1 + X_i \cdot i)\Big] \leq n + \sum_{i=3}^{n} \mathrm{E}[X_i] \cdot i \leq n + \sum_{i=3}^{n} i \cdot \Pr[X_i = 1] \leq n + \sum_{i=3}^{n} i \cdot \frac{2}{i} \leq 3n$, by linearity of expectation and since $\mathrm{E}[X_i] = \Pr[X_i = 1] \leq 2/i$.
4. Expected running time of the algorithm is $O(\mathrm{E}[R]) = O(n)$. ∎

Expected linear time analysis...

Theorem

P: *set of n points in the plane. Compute the closest pair of* **P** *in expected linear time.*

Proof.

1. $X_i = 1 \iff \alpha_i \neq \alpha_{i-1}$.
2. Running time is proportional to $R = 1 + \sum_{i=3}^{n}(1 + X_i \cdot i)$.
3. $\mathbf{E}[R] = \mathbf{E}\Bigl[1 + \sum_{i=3}^{n}(1 + X_i \cdot i)\Bigr] \leq n + \sum_{i=3}^{n} \mathbf{E}[X_i] \cdot i \leq n + \sum_{i=3}^{n} i \cdot \Pr[X_i = 1] \leq n + \sum_{i=3}^{n} i \cdot \frac{2}{i} \leq 3n$, by linearity of expectation and since $\mathbf{E}[X_i] = \Pr[X_i = 1] \leq 2/i$.
4. Expected running time of the algorithm is $O(\mathbf{E}[R]) = O(n)$. ∎

Expected linear time analysis...

Theorem

P: *set of* n *points in the plane. Compute the closest pair of* **P** *in expected linear time.*

Proof.

1. $X_i = 1 \iff \alpha_i \neq \alpha_{i-1}$.
2. Running time is proportional to $R = 1 + \sum_{i=3}^{n}(1 + X_i \cdot i)$.
3. $\mathbf{E}[R] = \mathbf{E}\Big[1 + \sum_{i=3}^{n}(1 + X_i \cdot i)\Big] \leq n + \sum_{i=3}^{n} \mathbf{E}[X_i] \cdot i \leq n + \sum_{i=3}^{n} i \cdot \mathbf{Pr}[X_i = 1] \leq n + \sum_{i=3}^{n} i \cdot \frac{2}{i} \leq 3n$, by linearity of expectation and since $\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] \leq 2/i$.
4. Expected running time of the algorithm is $O(\mathbf{E}[R]) = O(n)$. ∎

Expected linear time analysis...

Theorem

P: *set of* n *points in the plane. Compute the closest pair of* **P** *in expected linear time.*

Proof.

1. $X_i = 1 \iff \alpha_i \neq \alpha_{i-1}$.
2. Running time is proportional to $R = 1 + \sum_{i=3}^{n}(1 + X_i \cdot i)$.
3. $\mathbf{E}[R] = \mathbf{E}\left[1 + \sum_{i=3}^{n}(1 + X_i \cdot i)\right] \leq n + \sum_{i=3}^{n} \mathbf{E}[X_i] \cdot i \leq n + \sum_{i=3}^{n} i \cdot \Pr[X_i = 1] \leq n + \sum_{i=3}^{n} i \cdot \frac{2}{i} \leq 3n$, by linearity of expectation and since $\mathbf{E}[X_i] = \Pr[X_i = 1] \leq 2/i$.
4. Expected running time of the algorithm is $O(\mathbf{E}[R]) = O(n)$. ■

Expected linear time analysis...

Theorem

P: *set of* n *points in the plane. Compute the closest pair of* P *in expected linear time.*

Proof.

1. $X_i = 1 \iff \alpha_i \neq \alpha_{i-1}$.
2. Running time is proportional to $R = 1 + \sum_{i=3}^{n}(1 + X_i \cdot i)$.
3. $\mathbf{E}[R] = \mathbf{E}\left[1 + \sum_{i=3}^{n}(1 + X_i \cdot i)\right] \leq n + \sum_{i=3}^{n} \mathbf{E}[X_i] \cdot i \leq n + \sum_{i=3}^{n} i \cdot \Pr[X_i = 1] \leq n + \sum_{i=3}^{n} i \cdot \frac{2}{i} \leq 3n$, by linearity of expectation and since $\mathbf{E}[X_i] = \Pr[X_i = 1] \leq 2/i$.
4. Expected running time of the algorithm is $O(\mathbf{E}[R]) = O(n)$. ■

Expected linear time analysis...

Theorem

P: *set of* n *points in the plane. Compute the closest pair of* **P** *in expected linear time.*

Proof.

1. $X_i = 1 \iff \alpha_i \neq \alpha_{i-1}$.
2. Running time is proportional to $R = 1 + \sum_{i=3}^{n}(1 + X_i \cdot i)$.
3. $\mathbf{E}[R] = \mathbf{E}\Big[1 + \sum_{i=3}^{n}(1 + X_i \cdot i)\Big] \leq n + \sum_{i=3}^{n} \mathbf{E}[X_i] \cdot i \leq n + \sum_{i=3}^{n} i \cdot \Pr[X_i = 1] \leq n + \sum_{i=3}^{n} i \cdot \frac{2}{i} \leq 3n$, by linearity of expectation and since $\mathbf{E}[X_i] = \Pr[X_i = 1] \leq 2/i$.
4. Expected running time of the algorithm is $O(\mathbf{E}[R]) = O(n)$. ■

□

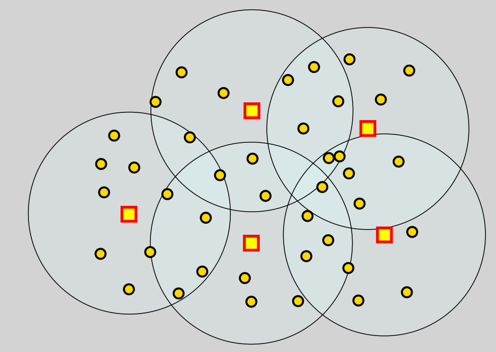
Part III

Computing nets

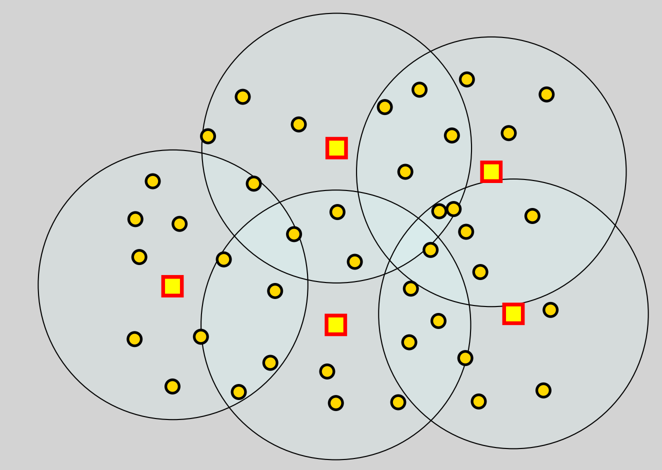
$\mathbf{r}$-net

$N \subseteq \mathbf{P}$ is an r-net if

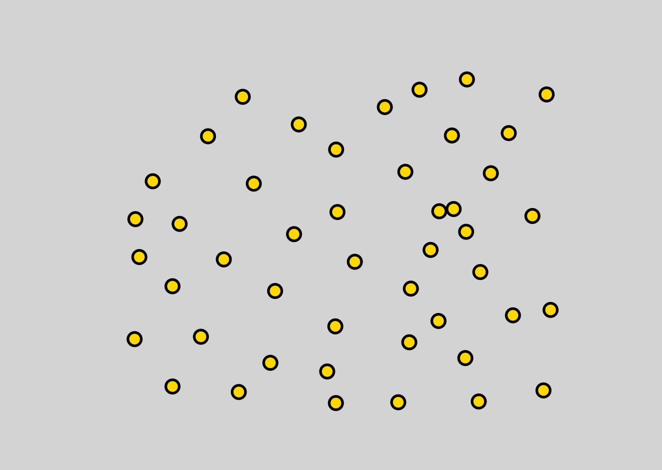
- Every point in $\mathbf{P}$ has distance $< r$ to a point in N
- For any two $\mathbf{p}, \mathbf{q} \in N$, we have $d(\mathbf{p}, \mathbf{q}) \geq r$.



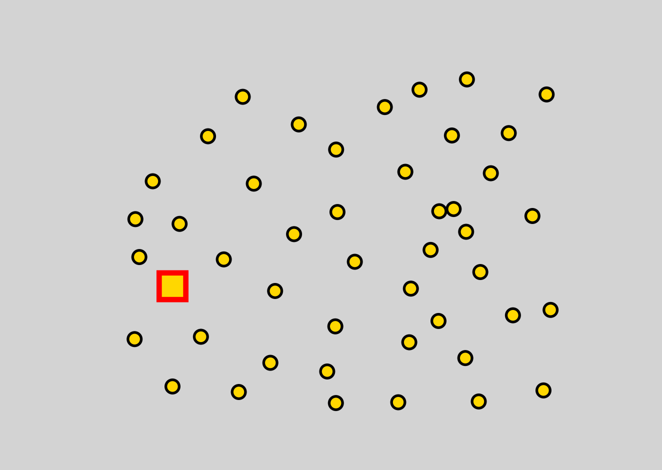
Computing an r-net



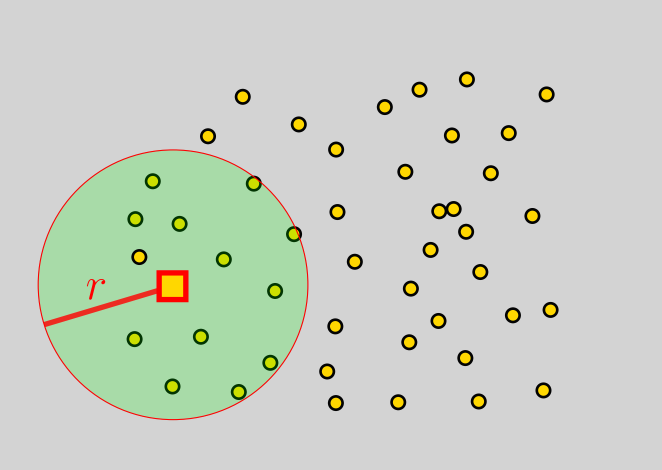
Computing an r-net



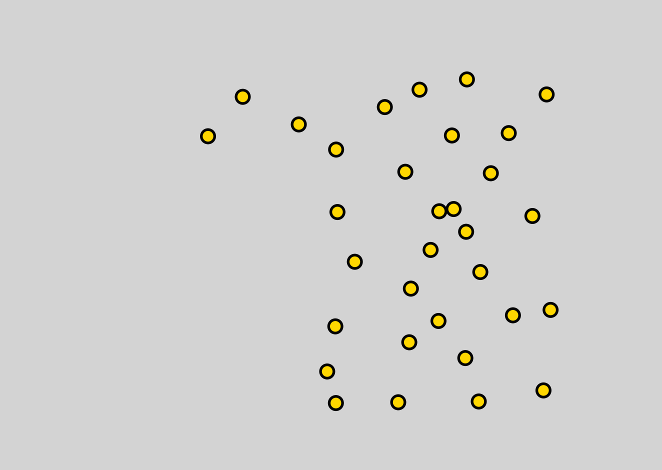
Computing an r-net



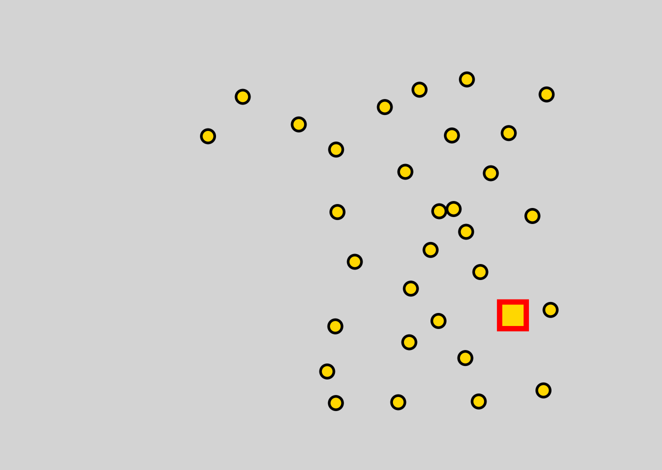
Computing an r-net



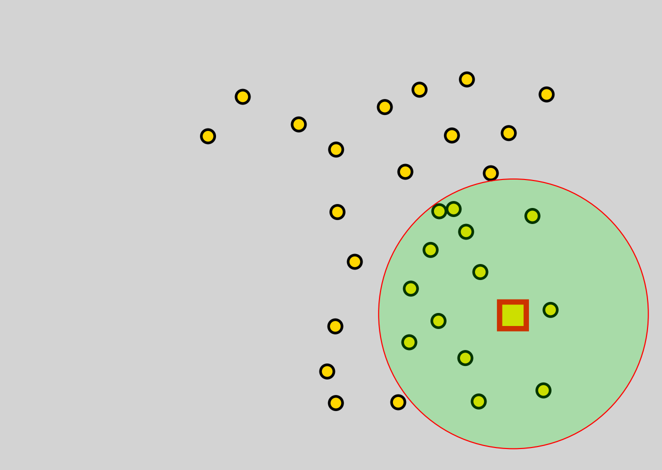
Computing an r-net



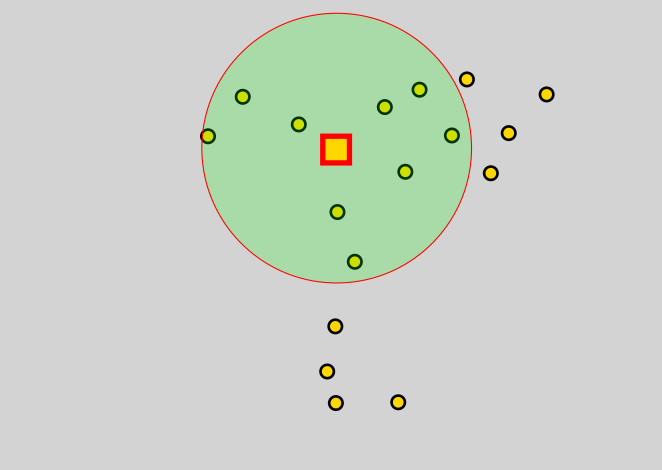
Computing an r-net



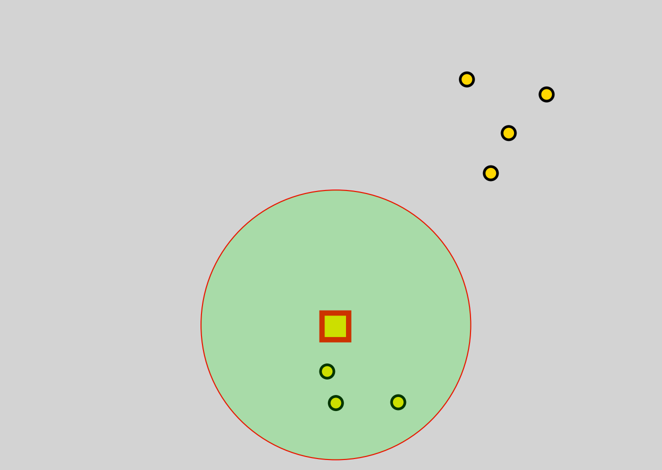
Computing an r-net



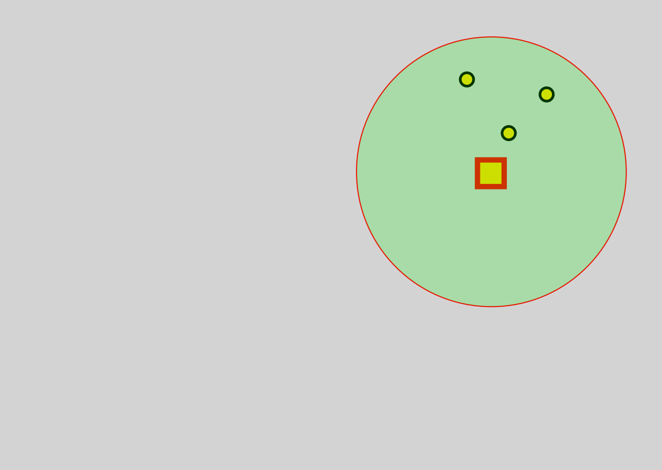
Computing an r-net



Computing an r-net

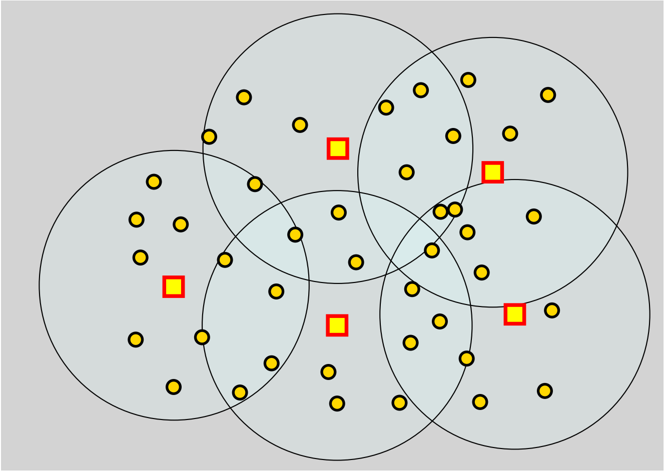


Computing an r-net



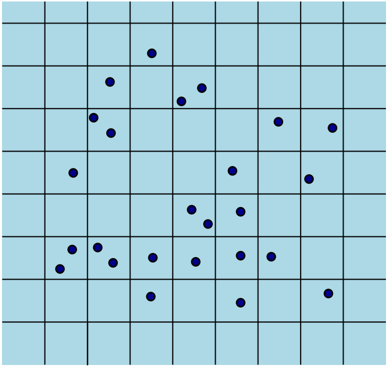
Computing an r-net

Computing an r-net



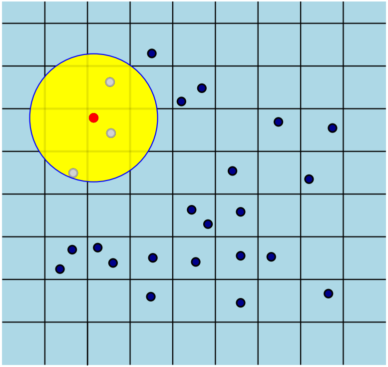
Application of Grids: Computing nets

...in linear time



Application of Grids: Computing nets

...in linear time



Repeatedly:

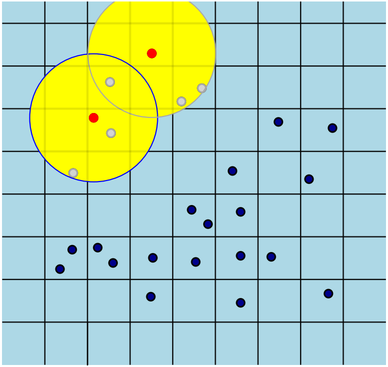
(1) Pick any unmarked point.
(2) Mark all neighbors in distance $< r$.

In an $\mathbf{r}$-grid

(A) Neighbors in distance $< r$, are in neighboring cells.
(B) Neighboring Cells found in $O(1)$ time.
(C) Cells contain lists of points.

Application of Grids: Computing nets

...in linear time



Repeatedly:

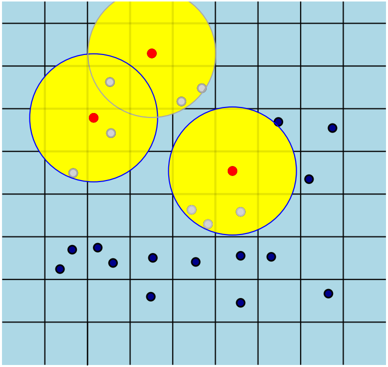
(1) Pick any unmarked point.
(2) Mark all neighbors in distance $< r$.

In an $\mathbf{r}$-grid

(A) Neighbors in distance $< r$, are in neighboring cells.
(B) Neighboring Cells found in $O(1)$ time.
(C) Cells contain lists of points.

Application of Grids: Computing nets

...in linear time



Repeatedly:

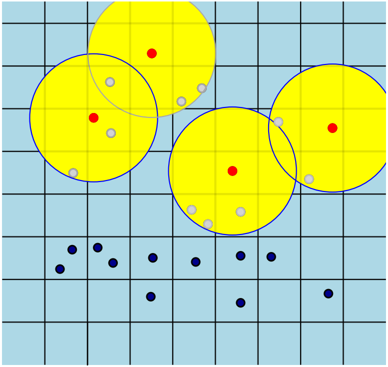
(1) Pick any unmarked point.
(2) Mark all neighbors in distance $< r$.

In an r-grid

(A) Neighbors in distance $< r$, are in neighboring cells.
(B) Neighboring Cells found in $O(1)$ time.
(C) Cells contain lists of points.

Application of Grids: Computing nets

...in linear time



Repeatedly:

(1) Pick any unmarked point.

(2) Mark all neighbors in distance $< r$.

In an r-grid

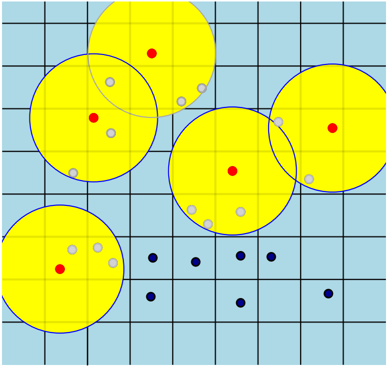
(A) Neighbors in distance $< r$, are in neighboring cells.

(B) Neighboring Cells found in $O(1)$ time.

(C) Cells contain lists of points.

Application of Grids: Computing nets

...in linear time



Repeatedly:

(1) Pick any unmarked point.

(2) Mark all neighbors in distance $< r$.

In an r-grid

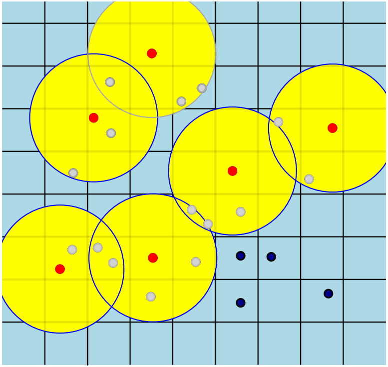
(A) Neighbors in distance $< r$, are in neighboring cells.

(B) Neighboring Cells found in $O(1)$ time.

(C) Cells contain lists of points.

Application of Grids: Computing nets

...in linear time



Repeatedly:

(1) Pick any unmarked point.

(2) Mark all neighbors in distance $< r$.

In an $\mathbf{r}$-grid

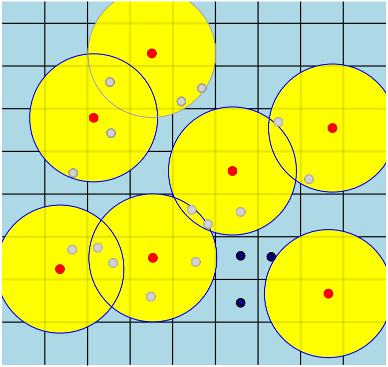
(A) Neighbors in distance $< r$, are in neighboring cells.

(B) Neighboring Cells found in $O(1)$ time.

(C) Cells contain lists of points.

Application of Grids: Computing nets

...in linear time



Repeatedly:

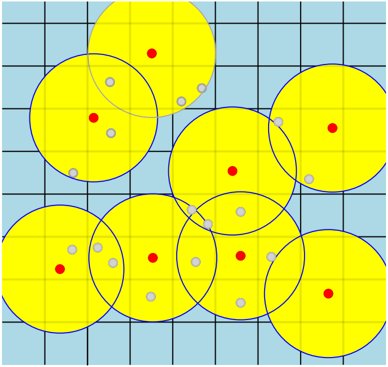
(1) Pick any unmarked point.
(2) Mark all neighbors in distance $< r$.

In an r-grid

(A) Neighbors in distance $< r$, are in neighboring cells.
(B) Neighboring Cells found in $O(1)$ time.
(C) Cells contain lists of points.

Application of Grids: Computing nets

...in linear time



Repeatedly:

(1) Pick any unmarked point.
(2) Mark all neighbors in distance $< r$.

In an r-grid

(A) Neighbors in distance $< r$, are in neighboring cells.
(B) Neighboring Cells found in $O(1)$ time.
(C) Cells contain lists of points.

Notes

Notes

Notes

Notes