Randomized Algorithms II – High Probability

Lecture 10 September 25, 2014

Part I

Movie...

Part II

Understanding the binomial distribution

Binomial distribution

 $X_n =$ numbers of heads when flipping a coin n times.

Claim

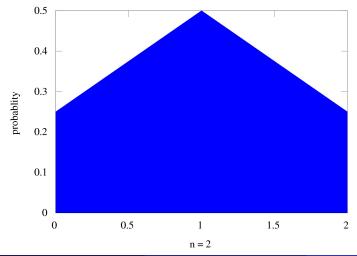
$$\Pr[X_n=i]=rac{\binom{n}{i}}{2^n}$$
.

Where: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.

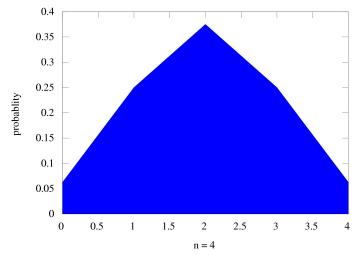
Indeed, $\binom{n}{i}$ is the number of ways to choose i elements out of n elements (i.e., pick which i coin flip come up heads).

Each specific such possibility (say 0100010...) had probability $1/2^n$.

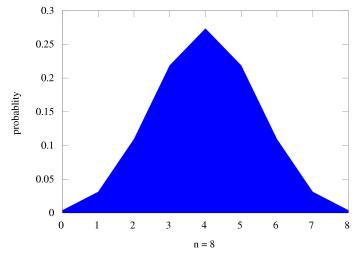
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



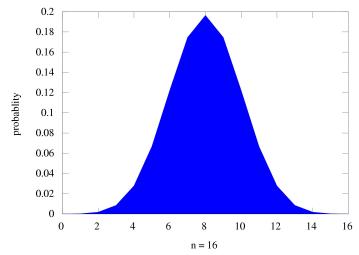
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



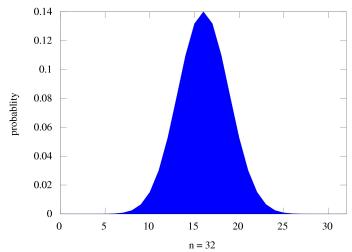
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



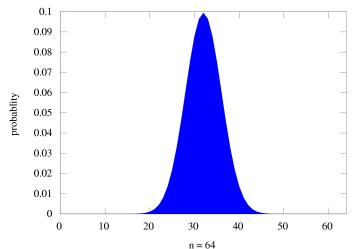
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



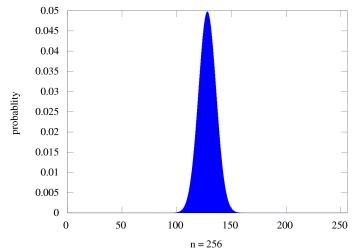
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



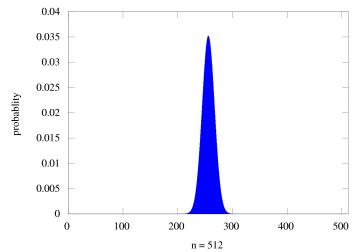
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



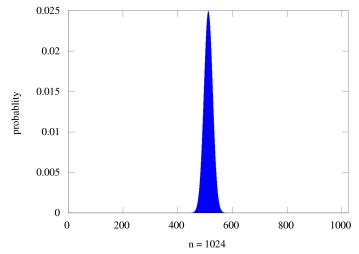
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



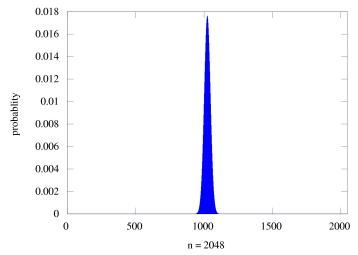
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



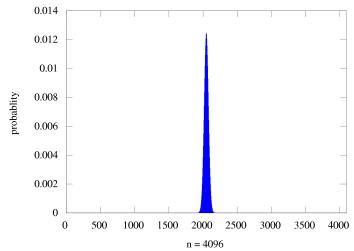
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



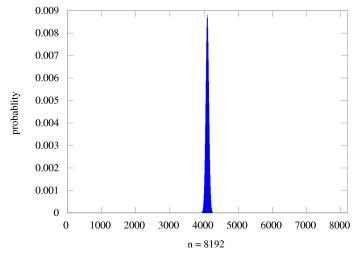
Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.

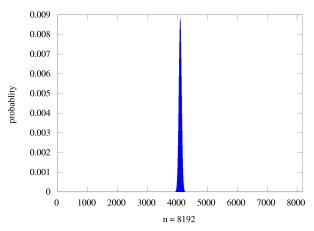


Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.



Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.





This is known as *concentration of mass*.

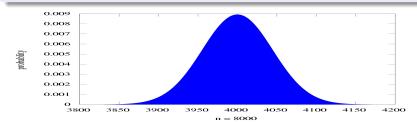
This is a very special case of the *law of large numbers*.

Side note...

Law of large numbers (weakest form)...

Informal statement of law of large numbers

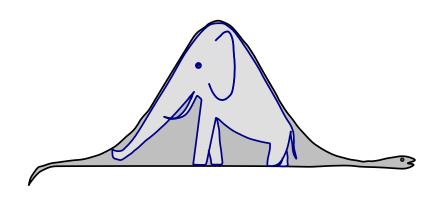
For n large enough, the middle portion of the binomial distribution looks like (converges to) the normal/Gaussian distribution.



Intuitive conclusion

Randomized algorithm are unpredictable in the tactical level, but very predictable in the strategic level.

What is really hiding below the Normal distribution?



Taken from ?.

Part III

QuickSort with high probability

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.
- 2 Track single element in input.
- Game ends, when this element is alone in subproblem
- ① Show every element in input, participates $\leq 32 \ln n$ rounds (with high enough probability).
- **6** \mathcal{E}_i : event *i*th element participates $> 32 \ln n$ rounds.
- \bigcirc Running time $O(C_{QS})$.

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.
- 2 Track single element in input.
- Game ends, when this element is alone in subproblem.
- ① Show every element in input, participates $\leq 32 \ln n$ rounds (with high enough probability).
- **6** \mathcal{E}_i : event *i*th element participates $> 32 \ln n$ rounds.
- \bigcirc Running time $O(C_{QS})$.

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.
- Track single element in input.
- Game ends, when this element is alone in subproblem.
- **3** Show every element in input, participates $\leq 32 \ln n$ rounds (with high enough probability).
- **1** \mathcal{E}_i : event *i*th element participates $> 32 \ln n$ rounds.
- extstyle ext

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.
- Track single element in input.
- Game ends, when this element is alone in subproblem.
- **3** Show every element in input, participates $\leq 32 \ln n$ rounds (with high enough probability).
- **5** \mathcal{E}_i : event *i*th element participates $> 32 \ln n$ rounds.
- ullet C_{QS} : number of comparisons performed by ${f QuickSort}$.
- $m{O}$ Running time $m{O}(m{C}_{QS})$

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.
- Track single element in input.
- Game ends, when this element is alone in subproblem.
- **Show every element in input, participates** $\leq 32 \ln n$ rounds (with high enough probability).
- **5** \mathcal{E}_i : event *i*th element participates $> 32 \ln n$ rounds.
- **©** C_{QS} : number of comparisons performed by **QuickSort**.
- $m{O}$ Running time $O(\mathit{C}_{\mathit{QS}})$

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.
- Track single element in input.
- Game ends, when this element is alone in subproblem.
- **3** Show every element in input, participates $\leq 32 \ln n$ rounds (with high enough probability).
- **5** \mathcal{E}_i : event *i*th element participates $> 32 \ln n$ rounds.
- **©** C_{QS} : number of comparisons performed by **QuickSort**.
- $m{0}$ Running time $m{O}(m{C}_{QS})$.
- $\begin{array}{l} \textbf{ OPRITY PROBLEM SETS STATE STATE$

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.
- Track single element in input.
- Game ends, when this element is alone in subproblem.
- Show every element in input, participates $\leq 32 \ln n$ rounds (with high enough probability).
- **5** \mathcal{E}_i : event *i*th element participates $> 32 \ln n$ rounds.
- **©** C_{QS} : number of comparisons performed by **QuickSort**.
- $oldsymbol{0}$ Running time $O(C_{QS})$.

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.
- Track single element in input.
- Game ends, when this element is alone in subproblem.
- **Show every element in input, participates** $\leq 32 \ln n$ rounds (with high enough probability).
- **5** \mathcal{E}_i : event *i*th element participates $> 32 \ln n$ rounds.
- **©** C_{QS} : number of comparisons performed by **QuickSort**.
- $m{0}$ Running time $m{O}(m{C}_{QS})$.

- $\begin{array}{l} \bullet \ \ \text{Probability of failure is} \\ \alpha = \Pr \big[C_{QS} \geq 32 n \ln n \big] \leq \Pr [\bigcup_i \mathcal{E}_i] \leq \sum_{i=1}^n \Pr \big[\mathcal{E}_i \big] \,. \end{array}$
- **2** Union bound: for any two events A and B: $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$.
- \bigcirc Assume: $\Pr[\mathcal{E}_i] \leq 1/n^3$.
- **9** Bad probability... $\alpha \leq \sum_{i=1}^n \Pr \left[\mathcal{E}_i \right] \leq \sum_{i=1}^n \frac{1}{n^3} = \frac{1}{n^2}$.
- \bigcirc QuickSort runs in $O(n \log n)$ time, with high probability.

- $\begin{array}{l} \bullet \ \ \text{Probability of failure is} \\ \alpha = \Pr \big[C_{QS} \geq 32 n \ln n \big] \leq \Pr [\bigcup_i \mathcal{E}_i] \leq \sum_{i=1}^n \Pr \big[\mathcal{E}_i \big] \,. \end{array}$
- **②** Union bound: for any two events A and B: $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$.
- \bullet Assume: $\Pr[\mathcal{E}_i] \leq 1/n^3$.
- ① Bad probability... $\alpha \leq \sum_{i=1}^n \Pr \left[\mathcal{E}_i \right] \leq \sum_{i=1}^n \frac{1}{n^3} = \frac{1}{n^2}$.
- \bigcirc \Longrightarrow QuickSort performs $\le 32n \ln n$ comparisons, w.h.p.
- \bigcirc QuickSort runs in $O(n \log n)$ time, with high probability.

- $\begin{array}{l} \bullet \ \ \text{Probability of failure is} \\ \alpha = \Pr \big[C_{QS} \geq 32 n \ln n \big] \leq \Pr [\bigcup_i \mathcal{E}_i] \leq \sum_{i=1}^n \Pr \big[\mathcal{E}_i \big] \,. \end{array}$
- **2** Union bound: for any two events A and B: $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$.
- **3** Assume: $\Pr[\mathcal{E}_i] \leq 1/n^3$.
- **3** Bad probability... $\alpha \leq \sum_{i=1}^n \Pr \left[\mathcal{E}_i \right] \leq \sum_{i=1}^n \frac{1}{n^3} = \frac{1}{n^2}$.
- ullet QuickSort performs $\leq 32n \ln n$ comparisons, w.h.p
- lacksquare QuickSort runs in $O(n \log n)$ time, with high probability.

- Probability of failure is $\left| lpha = \Pr \left| C_{QS} \geq 32 n \ln n
 ight| \leq \Pr [\bigcup_i \mathcal{E}_i] \leq \sum_{i=1}^n \Pr \Big[\mathcal{E}_i \Big]$.
- **2 Union bound**: for any two events A and B: $\Pr[A \cup B] < \Pr[A] + \Pr[B].$
- 3 Assume: $\Pr[\mathcal{E}_i] < 1/n^3$.
- **1** Bad probability... $\alpha \leq \sum_{i=1}^{n} \Pr[\mathcal{E}_i] \leq \sum_{i=1}^{n} \frac{1}{n^3} = \frac{1}{n^2}$.
- $\bullet \implies \mathsf{QuickSort}$ performs $< 32n \ln n$ comparisons, w.h.p.
- \implies QuickSort runs in $O(n \log n)$ time, with high

CS573 Fall 2014 12 / 45

- $\begin{array}{l} \bullet \ \ \text{Probability of failure is} \\ \alpha = \Pr \big[C_{QS} \geq 32 n \ln n \big] \leq \Pr [\bigcup_i \mathcal{E}_i] \leq \sum_{i=1}^n \Pr \big[\mathcal{E}_i \big] \,. \end{array}$
- **2** Union bound: for any two events A and B: $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$.
- 3 Assume: $\Pr[\mathcal{E}_i] < 1/n^3$.
- **3** Bad probability... $\alpha \leq \sum_{i=1}^n \Pr \left[\mathcal{E}_i \right] \leq \sum_{i=1}^n \frac{1}{n^3} = \frac{1}{n^2}$.
- \bigcirc \Longrightarrow QuickSort performs $\le 32n \ln n$ comparisons, w.h.p.
- lacksquare QuickSort runs in $O(n \log n)$ time, with high probability.

Proving that an element...

.. participates in small number of rounds.

- n: number of elements in input for QuickSort.
- 2 x: Arbitrary element x in input.
- $lackbox{0}{} S_i$: input to ith level recursive call that include x.
- $Y_i = 1 \iff x$ lucky in jth iteration.
- **Observation:** Y_1, Y_2, \ldots, Y_m are independent variables.
- ① x can participate $\leq
 ho = \log_{4/3} n \leq 3.5 \ln n$ rounds.
- $_{00}$...since $|S_j| \leq n(3/4)^{\# ext{ of lucky iteration in} 1...j}$
- $exttt{ } exttt{ }$

Proving that an element...

.. participates in small number of rounds.

- $oldsymbol{0}$ n: number of elements in input for **QuickSort**.
- 2 x: Arbitrary element x in input.
- $lackbox{0}{} S_i$: input to ith level recursive call that include x.
- § x lucky in jth iteration, if balanced split... $|S_{j+1}| \leq (3/4)\,|S_j|$ and $|S_j\setminus S_{j+1}| \leq (3/4)\,|S_j|$
- **6** $Y_i = 1 \iff x$ lucky in jth iteration.
- $\mathbf{O} \ \Pr \big[Y_j \big] = \frac{1}{2}.$
- **Observation:** Y_1, Y_2, \ldots, Y_m are independent variables.
- $m{0} \; \; x$ can participate $\leq
 ho = \log_{4/3} n \leq 3.5 \ln n$ rounds.
- $_{00}$...since $|S_{j}| \leq n(3/4)^{\# ext{ of lucky iteration in} 1...j}$
- $exttt{ } exttt{ }$

- n: number of elements in input for QuickSort.
- 2 x: Arbitrary element x in input.
- $lackbox{0}$ S_i : input to ith level recursive call that include x.
- $m{0}$ x **lucky** in jth iteration, if balanced split... $|S_{j+1}| \leq (3/4)\,|S_j|$ and $|S_j\setminus S_{j+1}| \leq (3/4)\,|S_j|$

- **Observation**: Y_1, Y_2, \ldots, Y_m are independent variables.
- ① x can participate $\leq
 ho = \log_{4/3} n \leq 3.5 \ln n$ rounds.
- $_{0}$...since $|S_{i}| \leq n(3/4)^{\# ext{ of lucky iteration in} 1...j}$.
- @ If ho lucky rounds in first k rounds $\implies |S_k| \leq (3/4)^{
 ho} n \leq 1.$

.. participates in small number of rounds.

- n: number of elements in input for QuickSort.
- 2 x: Arbitrary element x in input.
- **4** S_i : input to *i*th level recursive call that include x.
- $m{0}$ x **lucky** in jth iteration, if balanced split... $|S_{j+1}| \leq (3/4)\,|S_j|$ and $|S_j\setminus S_{j+1}| \leq (3/4)\,|S_j|$

- **Observation**: Y_1, Y_2, \ldots, Y_m are independent variables.
- ① x can participate $\leq
 ho = \log_{4/3} n \leq 3.5 \ln n$ rounds.
- lacksquare ...since $|S_i| \leq n(3/4)^{\# ext{ of lucky iteration in} 1...j}$.
- **1** If ho lucky rounds in first k rounds $\implies |S_k| \leq (3/4)^{
 ho} n \leq 1$.

Sariel (UIUC) CS573 13 Fall 2014 13 / 45

- Brain reset!
- ② Q: How many rounds x participates in = how many coin flips till one gets ρ heads?
- \odot A: In expectation, 2
 ho times.

- Brain reset!
- **2** Q: How many rounds x participates in = how many coin flips till one gets ρ heads?
- \odot A: In expectation, 2
 ho times.

- Brain reset!
- **2** Q: How many rounds x participates in = how many coin flips till one gets ρ heads?
- **3** A: In expectation, 2ρ times.

... participates in small number of rounds.

Assume the following:

Lemma

In M coin flips: $\Pr[\# \text{ heads} \le M/4] \le \exp(-M/8)$.

- ② Set $M = 32 \ln n \ge 8\rho$.

- \bigcirc probability $\leq
 ho \leq M/4$ ones in Y_1,\ldots,Y_M is

$$\leq \exp\left(-\frac{M}{8}\right) \leq \exp(-\rho) \leq \frac{1}{n^3}.$$

 $lackbox{}{}$ \Longrightarrow probability x participates in M recursive calls of $\mathsf{QuickSort} \leq 1/n^3$.

.. participates in small number of rounds.

Assume the following:

Lemma

In M coin flips: $\Pr[\# \text{ heads} \le M/4] \le \exp(-M/8)$.

- ② Set $M=32\ln n \geq 8
 ho$.
- $\Pr[Y_j = 0] = \Pr[Y_j = 1] = 1/2.$
- lacksquare Y_1, Y_2, \ldots, Y_M are independent.
- lacksquare probability $\leq
 ho \leq M/4$ ones in Y_1,\ldots,Y_M is

$$\leq \exp\left(-\frac{M}{8}\right) \leq \exp(-\rho) \leq \frac{1}{n^3}.$$

 \implies probability x participates in M recursive calls of QuickSort $\le 1/n^3$.

.. participates in small number of rounds.

- n input elements. Probability depth of recursion in QuickSort $> 32 \ln n$ is $< (1/n^3) * n = 1/n^2$.
- 2 Result:

Theorem

With high probability (i.e., $1 - 1/n^2$) the depth of the recursion of **QuickSort** is $\leq 32 \ln n$. Thus, with high probability, the running time of **QuickSort** is $O(n \log n)$.

Same result holds for MatchNutsAndBolts.

... participates in small number of rounds.

- n input elements. Probability depth of recursion in QuickSort $> 32 \ln n$ is $< (1/n^3) * n = 1/n^2$.
- 2 Result:

Theorem

With high probability (i.e., $1 - 1/n^2$) the depth of the recursion of QuickSort is $\leq 32 \ln n$. Thus, with high probability, the running time of QuickSort is $O(n \log n)$.

Same result holds for MatchNutsAndBolts.

.. participates in small number of rounds.

- *n* input elements. Probability depth of recursion in QuickSort $> 32 \ln n$ is $< (1/n^3) * n = 1/n^2$.
- Result:

Theorem

With high probability (i.e., $1 - 1/n^2$) the depth of the recursion of QuickSort is $\leq 32 \ln n$. Thus, with high probability, the running time of QuickSort is $O(n \log n)$.

Same result holds for MatchNutsAndBolts.

Alternative proof of high probability of **QuickSort**

- T: n items to be sorted.
- $\mathbf{Q} \quad t \in T$: element.
- **3** X_i : the size of subproblem in *i*th level of recursion containing t.
- \bullet $X_0 = n$, and $\mathbb{E}[X_i \mid X_{i-1}] \leq \frac{1}{2} \frac{3}{4} X_{i-1} + \frac{1}{2} X_{i-1} \leq \frac{7}{8} X_{i-1}$.
- $lackbox{ } orall ext{ random variables } ext{E}igg[Xigg] = ext{E}_yigg[ext{E}igg[Xigg|Y=yigg]igg].$
- $\bullet \ \operatorname{E}\big[X_i\big] = \operatorname{E}_y \bigg[\operatorname{E}\big[X_i \ \big| \ X_{i-1} = y \,\big] \,\bigg] \leq \operatorname{E}_{X_{i-1} = y} \big[\tfrac{7}{8}y\big] =$ $\frac{7}{8} \operatorname{E}[X_{i-1}] \leq \left(\frac{7}{8}\right)^i \operatorname{E}[X_0] = \left(\frac{7}{8}\right)^i n.$

CS573 Fall 2014 17 / 45

Alternative proof of high probability of **QuickSort**

- $lackbox{1}{\bullet} M = 8 \log_{8/7} n$: $\mu = \mathrm{E} ig[X_M ig] \leq ig(rac{7}{8} ig)^M n \leq rac{1}{n^8} n = rac{1}{n^7}$.
- ullet Markov's Inequality: For a non-negative variable X, and t>0, we have:

$$\Prig[X \geq tig] \quad \leq \quad rac{\mathrm{E}[X]}{t}.$$

By Markov's inequality:

$$\Pr \left[\begin{array}{c} t \text{ participates} \\ > M \text{ recursive calls} \end{array} \right] \leq \Pr \bigl[X_M \geq 1 \bigr] \leq \frac{\mathrm{E}[X_M]}{1} \leq \frac{1}{n^7}.$$

① Probability any element of input participates > M recursive calls $< n(1/n^7) < 1/n^6$.

Sariel (UIUC) CS573 18 Fall 2014 18 / 45

Part IV

Chernoff inequality

Preliminaries

1 X, Y: Random variables are independent if $\forall x, y$:

$$\Prigl[(X=x)\cap (Y=y)igr]=\Prigl[X=xigr]\cdot\Prigl[Y=yigr]\,.$$

The following is easy to prove:

Claim

If X and Y are independent

$$\implies E[XY] = E[X] E[Y].$$

$$\implies$$
 $Z = e^X$ and $W = e^Y$ are independent.

Chernoff inequality

Theorem (Chernoff inequality)

 X_1,\ldots,X_n : n independent random variables, such that $\Pr[X_i=1]=\Pr[X_i=-1]=rac{1}{2}$, for $i=1,\ldots,n$. Let $Y=\sum_{i=1}^n X_i$. Then, for any $\Delta>0$, we have

$$ext{Pr}ig[Y \geq \Deltaig] \leq ext{exp}ig(-\Delta^2/2nig)$$
 .

Fix arbitrary t > 0:

$$ext{Pr}ig[Y \geq \Deltaig] = ext{Pr}ig[tY \geq t\Deltaig]$$

Fix arbitrary t > 0:

$$ext{Pr}ig[Y \geq \Deltaig] = ext{Pr}ig[tY \geq t\Deltaig] = ext{Pr}ig[\exp(tY) \geq \exp(t\Delta)ig]$$

Sariel (UIUC) CS573 22 Fall 2014 22 / 45

Fix arbitrary t > 0:

$$egin{split} \Prig[Y \geq \Deltaig] &= \Prig[tY \geq t\Deltaig] &= \Prig[\exp(tY) \geq \exp(t\Delta)ig] \ &\leq rac{\mathrm{E}ig[\exp(tY)ig]}{\exp(t\Delta)}, \end{split}$$

$$\mathrm{E} \Big[\mathrm{exp}(t X_i) \Big] = rac{1}{2} e^t + rac{1}{2} e^{-t}.$$

$$\mathrm{E} \Big[\mathrm{exp}(tX_i) \Big] = rac{1}{2} e^t + rac{1}{2} e^{-t} = rac{e^t + e^{-t}}{2}.$$

$$\mathrm{E} \Big[\mathrm{exp}(tX_i) \Big] = rac{e^t + e^{-t}}{2}.$$

$$egin{aligned} \mathrm{E} \Big[\exp(t X_i) \Big] &= rac{e^t + e^{-t}}{2} \ &= rac{1}{2} \Big(1 + rac{t}{1!} + rac{t^2}{2!} + rac{t^3}{3!} + \cdots \Big) \ &+ rac{1}{2} \Big(1 - rac{t}{1!} + rac{t^2}{2!} - rac{t^3}{3!} + \cdots \Big) \,. \end{aligned}$$

$$ext{E} \Big[\exp(tX_i) \Big] = rac{1}{2} igg(1 + rac{t}{1!} + rac{t^2}{2!} + rac{t^3}{3!} + \cdots igg) \ + rac{1}{2} igg(1 - rac{t}{1!} + rac{t^2}{2!} - rac{t^3}{3!} + \cdots igg) \,.$$

$$egin{aligned} \mathrm{E} \Big[\exp(t X_i) \Big] &= rac{1}{2} igg(1 + rac{t}{1!} + rac{t^2}{2!} + rac{t^3}{3!} + \cdots igg) \ &+ rac{1}{2} igg(1 - rac{t}{1!} + rac{t^2}{2!} - rac{t^3}{3!} + \cdots igg) \ &= 1 + rac{t^2}{2!} + + \cdots + rac{t^{2k}}{(2k)!} + \cdots \,. \end{aligned}$$

$$\mathrm{E}\!\left[\exp(tX_i)
ight] = 1 + rac{t^2}{2!} + + \cdots + rac{t^{2k}}{(2k)!} + \cdots.$$

Continued...

$$\mathrm{E}\!\left[\exp(tX_i)
ight] = 1 + rac{t^2}{2!} + + \cdots + rac{t^{2k}}{(2k)!} + \cdots.$$

However: $(2k)! = k!(k+1)(k+2) \cdots 2k \ge k!2^k$.

Continued...

$$\mathrm{E}\!\left[\exp(tX_i)
ight] = 1 + rac{t^2}{2!} + + \cdots + rac{t^{2k}}{(2k)!} + \cdots.$$

However: $(2k)! = k!(k+1)(k+2)\cdots 2k \ge k!2^k$.

$$\mathrm{E}\!\left[\exp(tX_i)
ight] \,=\, \sum_{i=0}^\infty rac{t^{2i}}{(2i)!}$$

Continued...

$$\mathrm{E}\!\left[\exp(tX_i)
ight] = 1 + rac{t^2}{2!} + + \cdots + rac{t^{2k}}{(2k)!} + \cdots.$$

However: $(2k)! = k!(k+1)(k+2)\cdots 2k \ge k!2^k$.

$$ext{E} \Big[ext{exp}(tX_i) \Big] \, = \, \sum_{i=0}^\infty rac{t^{2i}}{(2i)!} \leq \sum_{i=0}^\infty rac{t^{2i}}{2^i(i!)}$$

$$ext{E} \Big[ext{exp}(tX_i) \Big] \, \leq \sum_{i=0}^{\infty} rac{t^{2i}}{2^i(i!)}$$

$$ext{E} \Big[ext{exp}(tX_i) \Big] \, \leq \sum_{i=0}^\infty rac{t^{2i}}{2^i(i!)} = \, \sum_{i=0}^\infty rac{1}{i!} \Big(rac{t^2}{2}\Big)^i$$

$$ext{E} \Big[ext{exp}(tX_i) \Big] \, \leq \sum_{i=0}^{\infty} rac{1}{i!} \Big(rac{t^2}{2}\Big)^i$$

Continued...

$$ext{E} \Big[ext{exp}(tX_i) \Big] \, \leq \sum_{i=0}^\infty rac{1}{i!} \Big(rac{t^2}{2}\Big)^i \, = \, ext{exp} \Big(rac{t^2}{2}\Big) \, .$$

Sariel (UIUC) CS573 23 Fall 2014 23 / 45

$$\mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$\mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$\operatorname{E}\!\left[\!\exp(tY)
ight] = \operatorname{E}\!\left[\exp\!\left(\sum_i tX_i
ight)
ight]$$

$$\mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$ext{E} \Big[ext{exp}(tY) \Big] = ext{E} \Big[ext{exp} \Big(\sum_i tX_i \Big) \Big] \ = ext{E} \Big[\prod_i ext{exp}(tX_i) \Big]$$

$$\mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$\mathrm{E}\!\left[\!\exp(tY)
ight] = \mathrm{E}\!\left[\prod_i \exp(tX_i)
ight]$$

Continued...

$$\mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$\mathrm{E}\!\left[\!\exp(tY)
ight] = \mathrm{E}\!\left[\prod_{i} \exp(tX_{i})
ight] \,=\, \prod_{i=1}^{n} \mathrm{E}\!\left[\!\exp(tX_{i})
ight]$$

Sariel (UIUC) CS573 23 Fall 2014 23 / 45

$$\mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$\mathrm{E} \Big[\! \exp(tY) \Big] = \prod_{i=1}^n \mathrm{E} \Big[\! \exp(tX_i) \Big]$$

Continued...

$$\mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$\mathrm{E}\!\left[\!\exp(tY)
ight] = \prod_{i=1}^n \mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \prod_{i=1}^n \exp\!\left(rac{t^2}{2}
ight)$$

Sariel (UIUC) CS573 23 Fall 2014 23 / 45

$$\mathrm{E}\!\left[\!\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$\mathrm{E}\!\left[\!\exp(tY)
ight] \leq \prod\limits_{i=1}^n \exp\!\left(rac{t^2}{2}
ight)$$

$$\operatorname{E}\!\left[\exp(tX_i)
ight] \, \leq \exp\!\left(rac{t^2}{2}
ight).$$

$$\operatorname{E}\!\left[\exp(tY)
ight] \leq \prod\limits_{i=1}^n \exp\!\left(rac{t^2}{2}
ight) \,=\, \exp\!\left(rac{nt^2}{2}
ight).$$

$$ext{E} \Big[ext{exp}(tY) \Big] \leq \prod_{i=1}^n ext{exp} \Big(rac{t^2}{2} \Big) \, = \, ext{exp} \Big(rac{nt^2}{2} \Big) \, .$$

$$\mathrm{E}\!\left[\exp(tY)
ight] \leq \exp\!\left(rac{nt^2}{2}
ight).$$

$$\operatorname{E}\!\left[\exp(tY)
ight] \leq \exp\!\left(rac{nt^2}{2}
ight).$$

$$\Pr[Y \geq \Delta]$$

$$\operatorname{E}\!\left[\exp(tY)
ight] \leq \exp\!\left(rac{nt^2}{2}
ight).$$

$$ext{Pr}ig[Y \geq \Deltaig] \leq rac{ ext{E}ig[ext{exp}(tY)ig]}{ ext{exp}(t\Delta)}$$

$$\operatorname{E}\!\left[\exp(tY)
ight] \leq \exp\!\left(rac{nt^2}{2}
ight).$$

$$ext{Pr}ig[Y \geq \Deltaig] \, \leq rac{ ext{E}\Big[ext{exp}(tY)\Big]}{ ext{exp}(t\Delta)} \leq rac{ ext{exp}\Big(rac{nt^2}{2}\Big)}{ ext{exp}(t\Delta)}$$

$$\operatorname{E}\!\left[\exp(tY)
ight] \leq \exp\!\left(rac{nt^2}{2}
ight).$$

$$ext{Pr}ig[Y \geq \Deltaig] \leq rac{ ext{exp}ig(rac{nt^2}{2}ig)}{ ext{exp}(t\Delta)}$$

Continued...

$$\operatorname{\mathbb{E}}\!\left[\exp(tY)
ight] \leq \exp\!\left(rac{nt^2}{2}
ight).$$

$$ext{Pr}ig[Y \geq \Deltaig] \, \leq rac{\expig(rac{nt^2}{2}ig)}{\exp(t\Delta)} = \expigg(rac{nt^2}{2} - t\Deltaigg)\,.$$

Sariel (UIUC) CS573 23 Fall 2014 23 / 45

$$\mathrm{E}\!\left[\!\exp(tY)
ight] \leq \exp\!\left(rac{nt^2}{2}
ight).$$

$$ext{Pr}ig[Y \geq \Deltaig] \, = \expigg(rac{nt^2}{2} - t\Deltaigg)\,.$$

Continued...

$$ext{Pr}ig[Y \geq \Deltaig] \, = \expigg(rac{nt^2}{2} - t\Deltaigg)\,.$$

Set $t = \Delta/n$:

Continued...

$$ext{Pr}ig[Y \geq \Deltaig] \, = \expigg(rac{nt^2}{2} - t\Deltaigg)\,.$$

Set $t = \Delta/n$:

$$ext{Pr}ig[Y \geq \Deltaig] \leq \expigg(rac{n}{2}igg(rac{\Delta}{n}igg)^2 - rac{\Delta}{n}\Deltaigg) = \expigg(-rac{\Delta^2}{2n}igg)\,.$$

Sariel (UIUC) CS573 23 Fall 2014 23 / 45

Continued...

$$ext{Pr}ig[Y \geq \Deltaig] \, = \expigg(rac{nt^2}{2} - t\Deltaigg)\,.$$

Set $t = \Delta/n$:

$$ext{Pr}ig[Y \geq \Deltaig] \leq \expigg(rac{n}{2}igg(rac{\Delta}{n}igg)^2 - rac{\Delta}{n}\Deltaigg) = \expigg(-rac{\Delta^2}{2n}igg)\,.$$

Sariel (UIUC) CS573 23 Fall 2014 23 / 45

Chernoff inequality...

...what it really says

By theorem:

$$ext{Pr}ig[Y \geq \Deltaig] = \sum_{i=\Delta}^n ext{Pr}ig[Y = iig] = \sum_{i=n/2+\Delta/2}^n rac{inom{n}{i}}{2^n} \leq ext{exp}igg(-rac{\Delta^2}{2n}igg)\,,$$

Sariel (UIUC) CS573 24 Fall 2014 24 / 45

symmetry

Corollary

Let X_1,\ldots,X_n be n independent random variables, such that $\Pr[X_i=1]=\Pr[X_i=-1]=\frac{1}{2}$, for $i=1,\ldots,n$. Let $Y=\sum_{i=1}^n X_i$. Then, for any $\Delta>0$, we have

$$ext{Pr}ig[|Y| \geq \Deltaig] \leq 2 \expigg(-rac{\Delta^2}{2n}igg)\,.$$

Chernoff inequality for coin flips

 X_1, \ldots, X_n be *n* independent coin flips, such that $\Pr[X_i=1]=\Pr[X_i=0]=rac{1}{2}$, for $i=1,\ldots,n$. Let $Y = \sum_{i=1}^n X_i$. Then, for any $\Delta > 0$, we have

$$\Pr\Bigl[\frac{n}{2} - Y \geq \Delta\Bigr] \leq \exp\Bigl(-\frac{2\Delta^2}{n}\Bigr) \quad \text{ and } \quad \Pr\Bigl[Y - \frac{n}{2} \geq \Delta\Bigr] \leq$$

In particular, we have
$$\Pr\Big[Y-rac{n}{2}\Big] \geq \Delta\Big] \leq 2\exp\Big(-rac{2\Delta^2}{n}\Big)$$
 .

CS573 26 Fall 2014 26 / 45

The special case we needed

Lemma

In a sequence of M coin flips, the probability that the number of ones is smaller than $L \leq M/4$ is at most $\exp(-M/8)$.

Proof.

Let $Y = \sum_{i=1}^{m} X_i$ the sum of the M coin flips. By the above corollary, we have:

$$ext{Pr}ig[Y \leq Lig] = ext{Pr}igg[rac{M}{2} - Y \geq rac{M}{2} - Ligg] = ext{Pr}igg[rac{M}{2} - Y \geq \Deltaigg]\,,$$

where $\Delta = M/2 - L \geq M/4$. Using the above Chernoff inequality, we get

$$\Prig[Y \le Lig] \le \expig(-rac{2\Delta^2}{M}ig) \le \exp(-M/8).$$

Part V

The Chernoff Bound — General Case

The general problem

Problem

Let $X_1, \ldots X_n$ be n independent Bernoulli trials, where

$$ext{Pr}ig[X_i=1ig]=p_i \quad ext{ and } \quad ext{Pr}ig[X_i=0ig]=1-p_i,$$

and let denote

$$Y = \sum_i X_i \qquad \mu = \mathop{\mathrm{E}}[Y]$$
 .

Question: what is the probability that $Y \geq (1 + \delta)\mu$.

The Chernoff Bound

The general case

Theorem (Chernoff inequality)

For any $\delta > 0$,

$$ext{Pr}ig[Y>(1+\delta)\muig]$$

Or in a more simplified form, for any $\delta \leq 2e-1$,

$$\Pr\!\left[Y>(1+\delta)\mu\right]<\exp\!\left(-\mu\delta^2/4\right),$$

and

$$\Prig[Y > (1+\delta)\muig] < 2^{-\mu(1+\delta)},$$

for $\delta \geq 2e-1$.

Sariel (UIUC) CS573 30 Fall 2014 30 / 45

Theorem

Theorem

Under the same assumptions as the theorem above, we have

$$\Prig[Y < (1-\delta)\muig] \leq \expigg(-\murac{\delta^2}{2}igg)$$
 .

Part VI

Treaps

- Work usually by storing additional information.
- ② Idea: For every element x inserted randomly choose priority $p(x) \in [0,1]$.
- $X = \{x_1, \dots, x_n\}$ priorities: $p(x_1), \dots, p(x_n)$.
- **4** x_k : lowest priority in X.
- \bullet Make x_k the root.
- partition X in the natural way:
 - (A) L: set of all the numbers smaller than x_k in X, and
 - (B) R: set of all the numbers larger than x_k in X.

- Work usually by storing additional information.
- ② Idea: For every element x inserted randomly choose priority $p(x) \in [0,1]$.
- $X = \{x_1, \dots, x_n\}$ priorities: $p(x_1), \dots, p(x_n)$.
- $ext{ } ext{ } ext$
- Make x_k the root.
- \odot partition X in the natural way:
 - (A) L: set of all the numbers smaller than x_k in X, and
 - (B) R: set of all the numbers larger than x_k in X.

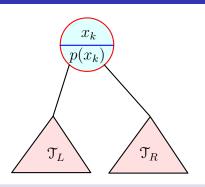
- Work usually by storing additional information.
- ② Idea: For every element x inserted randomly choose **priority** $p(x) \in [0,1]$.
- $X = \{x_1, \dots, x_n\}$ priorities: $p(x_1), \dots, p(x_n)$.
- \boldsymbol{a} \boldsymbol{x}_k : lowest priority in \boldsymbol{X} .
- \bullet Make x_k the root.
- partition in the natural way:
 - (A) L: set of all the numbers smaller than x_k in X, and
 - (B) R: set of all the numbers larger than x_k in X.

- Work usually by storing additional information.
- ② Idea: For every element x inserted randomly choose **priority** $p(x) \in [0,1]$.
- $X = \{x_1, \dots, x_n\}$ priorities: $p(x_1), \dots, p(x_n)$.
- **4** x_k : lowest priority in X.
- \bullet Make x_k the root.
- partition in the natural way:
 - (A) L: set of all the numbers smaller than x_k in X, and
 - (B) R: set of all the numbers larger than x_k in X.

- Work usually by storing additional information.
- ② Idea: For every element x inserted randomly choose **priority** $p(x) \in [0,1]$.
- $X = \{x_1, \dots, x_n\}$ priorities: $p(x_1), \dots, p(x_n)$.
- **4** x_k : lowest priority in X.
- **1** Make x_k the root.
- partition in the natural way:
 - (A) L: set of all the numbers smaller than x_k in X, and
 - (B) R: set of all the numbers larger than x_k in X

- Work usually by storing additional information.
- ② Idea: For every element x inserted randomly choose **priority** $p(x) \in [0,1]$.
- $X = \{x_1, \dots, x_n\}$ priorities: $p(x_1), \dots, p(x_n)$.
- **4** x_k : lowest priority in X.
- **10** Make x_k the root.
- - (A) L: set of all the numbers smaller than x_k in X, and
 - (B) R: set of all the numbers larger than x_k in X.

Treaps



Continuing recursively, we have:

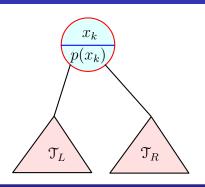
- (A) L: set of all the numbers smaller than x_k in X, and
- (B) R: set of all the numbers larger than x_k in X.

Definition

Resulting tree a *treap*

Tree over the elements, and a heap over the priorities; that is, TREAP = TREE + HEAP.

Treaps



Continuing recursively, we have:

- (A) L: set of all the numbers smaller than x_k in X, and
- (B) R: set of all the numbers larger than x_k in X.

Definition

Resulting tree a *treap*.

Tree over the elements, and a heap over the priorities; that is, TREAP = TREE + HEAP.

Treaps continued

Lemma

S: n elements.

Expected depth of treap **T** for S is $O(\log(n))$.

Depth of treap T for S is $O(\log(n))$ w.h.p.

Proof

QuickSort...

Treaps continued

Lemma

S: n elements.

Expected depth of treap **T** for S is $O(\log(n))$.

Depth of treap T for S is $O(\log(n))$ w.h.p.

Proof.

QuickSort...

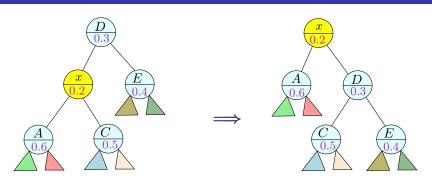


Treaps - implementation

Observation

Given n distinct elements, and their (distinct) priorities, the treap storing them is uniquely defined.

Rotate right...



Sariel (UIUC) CS573 37 Fall 2014 37 / 45

- \bullet x: an element x to insert.
- Insert it into T as a regular binary tree.
- \odot Takes $O(\text{height}(\mathbf{T}))$.
- $oldsymbol{0}$ x is a leaf in the treap
- lacksquare Pick priority $p(x) \in [0,1]$.
- lacktriangle Valid search tree,.. but priority heap is broken at $oldsymbol{x}$.
- \bigcirc Fix priority heap around x.

- \bullet x: an element x to insert.
- Insert it into T as a regular binary tree.
- \odot Takes $O(\text{height}(\mathbf{T}))$.
- $oldsymbol{0}$ x is a leaf in the treap
- lacksquare Pick priority $p(x) \in [0,1]$.
- lacktriangle Valid search tree,.. but priority heap is broken at $oldsymbol{x}$.
- \bigcirc Fix priority heap around x.

- \bullet x: an element x to insert.
- Insert it into T as a regular binary tree.
- **3** Takes $O(\text{height}(\mathbf{T}))$.
- $oldsymbol{0}$ $oldsymbol{x}$ is a leaf in the treap
- lacksquare Pick priority $p(x) \in [0,1]$.
- lacktriangle Valid search tree,.. but priority heap is broken at $oldsymbol{x}$.
- \bigcirc Fix priority heap around x.

- $\mathbf{0}$ \mathbf{x} : an element \mathbf{x} to insert.
- Insert it into T as a regular binary tree.
- **3** Takes $O(\text{height}(\mathbf{T}))$.
- $oldsymbol{0}$ x is a leaf in the treap.
- lacksquare Pick priority $p(x) \in [0,1]$.
- lacktriangle Valid search tree,.. but priority heap is broken at $oldsymbol{x}$.
- \bigcirc Fix priority heap around x.

- $\mathbf{0}$ \mathbf{x} : an element \mathbf{x} to insert.
- Insert it into T as a regular binary tree.
- **3** Takes $O(\text{height}(\mathbf{T}))$.
- $oldsymbol{0}$ x is a leaf in the treap.
- lacksquare Pick priority $p(x) \in [0,1]$.
- lacktriangle Valid search tree,.. but priority heap is broken at $oldsymbol{x}$.
- \bigcirc Fix priority heap around x.

- \bullet x: an element x to insert.
- Insert it into T as a regular binary tree.
- **3** Takes $O(\text{height}(\mathbf{T}))$.
- $oldsymbol{0}$ x is a leaf in the treap.
- lacksquare Pick priority $p(x) \in [0,1]$.
- \bullet Valid search tree,.. but priority heap is broken at x.
- \bigcirc Fix priority heap around x.

- $oldsymbol{0}$ $oldsymbol{x}$: an element $oldsymbol{x}$ to insert.
- Insert it into T as a regular binary tree.
- **3** Takes $O(\text{height}(\mathbf{T}))$.
- $oldsymbol{0}$ $oldsymbol{x}$ is a leaf in the treap.
- lacksquare Pick priority $p(x) \in [0,1]$.
- \bullet Valid search tree,.. but priority heap is broken at x.
- **o** Fix priority heap around x.

Fix treap for a leaf x...

```
\begin{aligned} & \text{RotateUp}(x) \\ & y \leftarrow \text{parent}(x) \\ & \text{while } p(y) > p(x) \text{ do} \\ & \text{if } y.\text{left\_child} = x \text{ then} \\ & \text{RotateRight}(y) \\ & \text{else} \\ & \text{RotateLeft}(y) \\ & y \leftarrow \text{parent}(x) \\ \\ & \text{Insertion takes } O(\text{height}(\mathsf{T})). \end{aligned}
```

Sariel (UIUC) CS573 39 Fall 2014 39 / 45

Fix treap for a leaf x...

```
\begin{aligned} & \text{RotateUp}(x) \\ & y \leftarrow \operatorname{parent}(x) \\ & \text{while } p(y) > p(x) \text{ do} \\ & \text{if } y.\operatorname{left\_child} = x \text{ then} \\ & \text{RotateRight}(y) \\ & \text{else} \\ & \text{RotateLeft}(y) \\ & y \leftarrow \operatorname{parent}(x) \\ & \text{Insertion takes } O(\operatorname{height}(\mathbf{T})). \end{aligned}
```

Sariel (UIUC) CS573 39 Fall 2014 39 / 45

- Deletion is just an insertion done in reverse.
- 2 x: element to delete.

- Solution Rotate so that child with lower priority becomes new parent.
- is now leaf deleting is easy...

- Deletion is just an insertion done in reverse.
- x: element to delete.

- Deletion is just an insertion done in reverse.
- 2 x: element to delete.
- lacktriangledown rotate x down till its a leaf.
- Rotate so that child with lower priority becomes new parent
- $oldsymbol{o}$ $oldsymbol{x}$ is now leaf deleting is easy...

- Deletion is just an insertion done in reverse.
- 2 x: element to delete.
- lacktriangledown rotate $oldsymbol{x}$ down till its a leaf.
- Solution
 Rotate so that child with lower priority becomes new parent.
- $lackbox{0} x$ is now leaf deleting is easy...

- Deletion is just an insertion done in reverse.
- 2 x: element to delete.
- lacktriangledown rotate $oldsymbol{x}$ down till its a leaf.
- Rotate so that child with lower priority becomes new parent.
- $oldsymbol{0}$ $oldsymbol{x}$ is now leaf deleting is easy...

- x: element stored in treap T.
- 2 split **T** into two treaps one treap $\mathbf{T}_{\leq x}$ and treap $\mathbf{T}_{>}$ for all the elements larger than x.
- fix priorities by rotation.
- Splitting is now easy....
- lacksquare Restore x to its original priority. Fix by rotations

- $oldsymbol{0}$ $oldsymbol{x}$: element stored in treap $oldsymbol{\mathsf{T}}$.
- ② split T into two treaps one treap $T_{\leq x}$ and treap $T_{>}$ for all the elements larger than x.
- fix priorities by rotation.
- Splitting is now easy....
- $ilde{o}$ Restore x to its original priority. Fix by rotations

- $oldsymbol{0}$ $oldsymbol{x}$: element stored in treap $oldsymbol{\mathsf{T}}$.
- 2 split T into two treaps one treap $T_{\leq x}$ and treap $T_{>}$ for all the elements larger than x.
- fix priorities by rotation.
- x item is now the root.
- Splitting is now easy....
- $ilde{o}$ Restore x to its original priority. Fix by rotations

- $oldsymbol{0}$ $oldsymbol{x}$: element stored in treap $oldsymbol{\mathsf{T}}$.
- ② split T into two treaps one treap $T_{\leq x}$ and treap $T_{>}$ for all the elements larger than x.
- fix priorities by rotation.
- Splitting is now easy....
- $oldsymbol{o}$ Restore $oldsymbol{x}$ to its original priority. Fix by rotations

- $oldsymbol{0}$ $oldsymbol{x}$: element stored in treap $oldsymbol{\mathsf{T}}$.
- 2 split T into two treaps one treap $T_{\leq x}$ and treap $T_{>}$ for all the elements larger than x.
- fix priorities by rotation.
- x item is now the root.
- Splitting is now easy....
- $ilde{o}$ Restore x to its original priority. Fix by rotations

- $oldsymbol{0}$ $oldsymbol{x}$: element stored in treap $oldsymbol{\mathsf{T}}$.
- ② split T into two treaps one treap $T_{\leq x}$ and treap $T_{>}$ for all the elements larger than x.
- fix priorities by rotation.
- x item is now the root.
- Splitting is now easy....
- $oldsymbol{\circ}$ Restore x to its original priority. Fix by rotations.

Meld

- \bullet T_L and T_R : treaps.
- ② all elements in T_L i all elements in T_R .
- Want to merge them into a single treap...

Sariel (UIUC) CS573 42 Fall 2014 42 / 45

Treap – summary

Theorem

Let T be an empty treap, after a sequence of $m=n^c$ insertions, where c is some constant.

d: arbitrary constant.

The probability depth **T** ever exceed $d \log n$ is $\leq 1/n^{O(1)}$.

A treap can handle insertion/deletion in $O(\log n)$ time with high probability.

- $\mathbf{0}$ $\mathbf{T}_1, \dots, \mathbf{T}_m$: sequence of treaps.
- $\Pr \Big[\operatorname{depth}(\mathsf{T}_i) > c' t \Big(rac{\log n}{\log |\mathsf{T}_i|} \Big) \cdot \log |\mathsf{T}_i| \Big] \leq rac{1}{n^{O(1)}},$



- $\mathbf{0}$ $\mathbf{T}_1, \dots, \mathbf{T}_m$: sequence of treaps.
- $\mathbf{Q} \mathbf{T}_i$ is treap after *i*th operation.
- $oldsymbol{0} |lpha_i = ext{Pr} \left| \operatorname{depth}(\mathsf{T}_i) > tc' \log n
 ight| =$ $\Pr\left[\operatorname{depth}(\mathsf{T}_i) > c't\left(\frac{\log n}{\log|\mathsf{T}_i|}\right) \cdot \log|\mathsf{T}_i|\right] \leq \frac{1}{n^{O(1)}},$



- **1** T_1, \ldots, T_m : sequence of treaps.
- $\begin{array}{l} \text{ @ } \alpha_i = \Pr \bigl[\operatorname{depth}(\mathsf{T}_i) > tc' \log n \bigr] = \\ \Pr \bigl[\operatorname{depth}(\mathsf{T}_i) > c't \bigl(\frac{\log n}{\log |\mathsf{T}_i|} \bigr) \cdot \log |\mathsf{T}_i| \bigr] \leq \frac{1}{n^{O(1)}}, \end{array}$
- Use union bound...



- **1** T_1, \ldots, T_m : sequence of treaps.
- $\mathbf{Q} \mathbf{T}_i$ is treap after ith operation.
- $\begin{array}{l} \boldsymbol{\Im} \ \ \alpha_i = \Pr \bigl[\operatorname{depth}(\mathsf{T}_i) > tc' \log n \bigr] = \\ \ \ \Pr \bigl[\operatorname{depth}(\mathsf{T}_i) > c't \Bigl(\frac{\log n}{\log |\mathsf{T}_i|} \Bigr) \cdot \log |\mathsf{T}_i| \Bigr] \leq \frac{1}{n^{O(1)}}, \end{array}$
- Use union bound...



Bibliographical Notes

- Chernoff inequality was a rediscovery of Bernstein inequality.
- 2 ...published in 1924 by Sergei Bernstein.
- Treaps were invented by Siedel and Aragon ?.
- Experimental evidence suggests that Treaps performs reasonably well in practice see ?.
- Old implementation of treaps I wrote in C is available here: http://valis.cs.uiuc.edu/blog/?p=6060.

Sariel (UIUC) CS573 46 Fall 2014 46 / 48

Sariel (UIUC) CS573 47 Fall 2014 47 / 49

Sariel (UIUC) CS573 48 Fall 2014 48 / 49

Sariel (UIUC) CS573 49 Fall 2014 49 / 45