L7: Advanced Dynamic programming

9/12/23

Optimal binary search trees

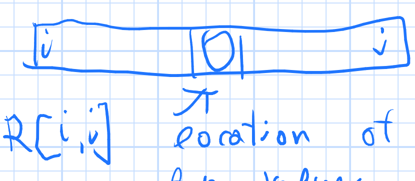
$f[1 \ldots n]$: frequencies

$$cost(T) = \sum_{i=1}^{n} depth_T(i) \cdot f[i]$$

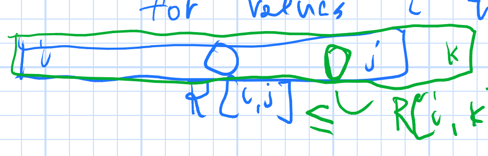
$c(i,j)$ = cost for tree for values $i \ldots j$

$$c(i,j) = \min_{k=i}^{j} \left(c(i,k-1) + \sum_{t=i}^{j} f[t] + c(k+1,j) \right)$$

$$O(n^2) \cdot O(n) = O(n^3)$$



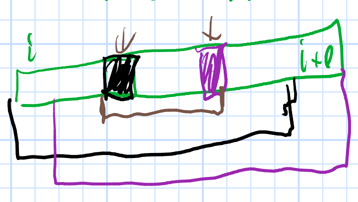
$R[i,j]$ location of root for optimal BST. for values i to j.



$k > j$

$R[i,j] \leq R[i,k]$

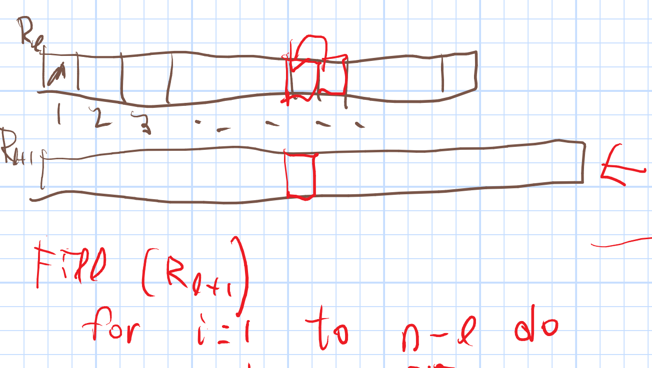
$R_\ell[i]$. – location of the root of optimal BST for the interval $[i \ldots i+\ell-1]$



$R_{\ell+1}[i]$

<u>Observation</u> opt root $R_{\ell+1}[i]$ is in $\overbrace{R_\ell[i] \ldots R_\ell[i+1]}^{k_n}$

Computing $R_{\ell+1}[i]$ takes $O(1 + R_\ell[i+1] - R_\ell[i])$



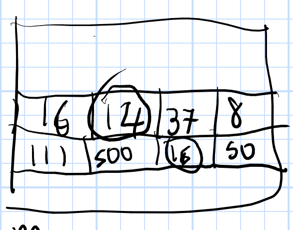
Fill $(R_{\ell+1})$
  for $i = 1$ to $n-\ell$ do
    compute $R_{\ell+1}[i]$

$$\sum_{i=1}^{n-\ell} O(1 + R_\ell[i+1] - R_\ell[i]) = O(n)$$

$O(n^2)$ overall

Find the minimum in a monotone matrix

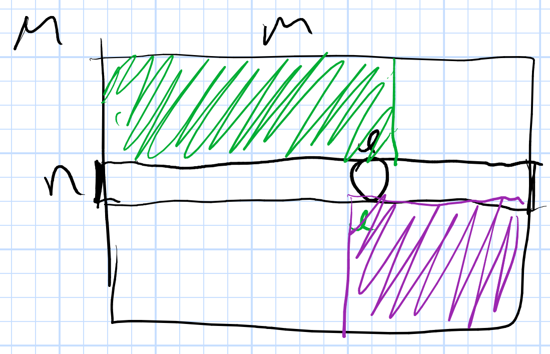
$M[n \times m]$ it is monotone



$L[i] = \arg\min_{j=1}^{m} M[i,j]$ column in ith row realizing the minimum.

<u>Task</u> compute $L[1 \ldots n]$

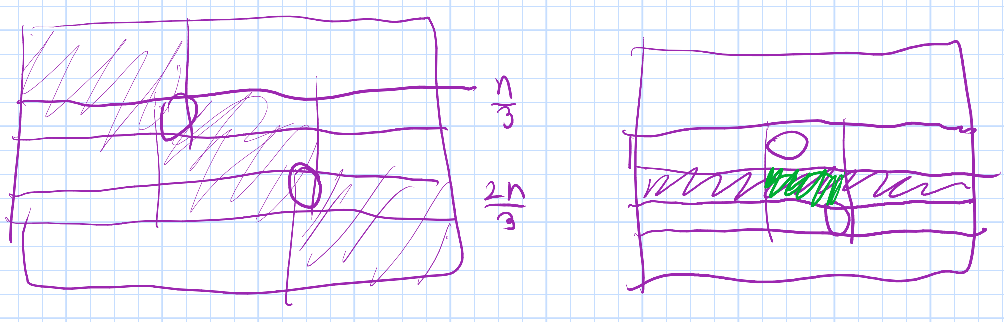
Naive algorithm scan every row, compute min, $O(nm)$



mid row, compute mi.

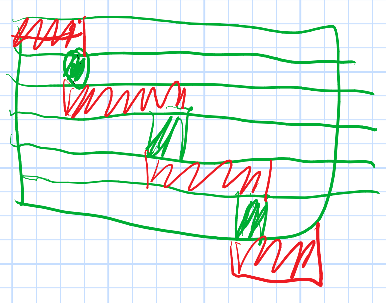
$$T(n,m) = O(m) + T\left(\frac{n}{2}, \ell\right) + T\left(\frac{n}{2}, m-\ell\right)$$

$$T(n,m) = O(n \log m)$$



Recursively compute min for all even rows.

$$T(n,m) = T\left(\frac{n}{2}, m\right)$$



$m + \frac{n}{2}$

$O(n+m)$

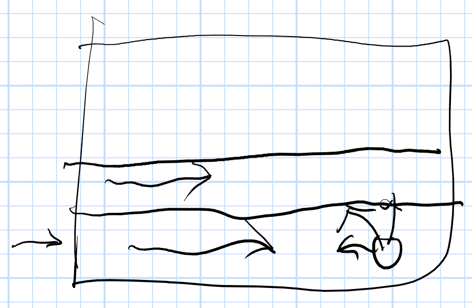
$$T(n,m) = T\left(\frac{n}{2}, m\right) + O(n+m)$$
$$= O((n+m)\log n)$$

---

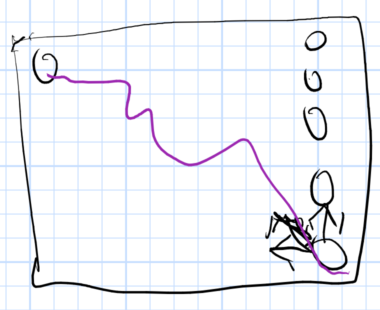
edit distance

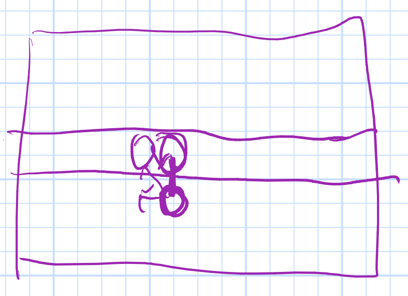
$A[1 \ldots n]$ $B[1 \ldots m]$

$$ed(i,j) = \min \begin{cases} ed(i-1,j)+1 \\ ed(i,j-1)+1 \\ ed(i-1,j-1) + [A[i] \neq B[j]] \end{cases}$$



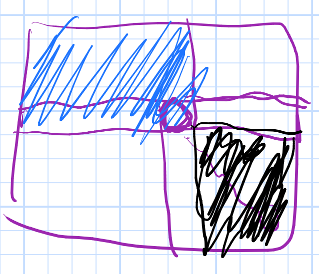
$O(mn)$ time space





$O(n+m)$ space

$O(nm)$ RT



$O(n+m)$

$$T(n,m) = \underset{c'}{O}(nm) + \max_{\ell} \left(\underbrace{T\left(\frac{n}{2}, \ell\right)}_{c\frac{n}{2}\ell} + \underbrace{T\left(\frac{n}{2}, m-\ell\right)}_{c\frac{n}{2}(m-\ell)} \right)$$

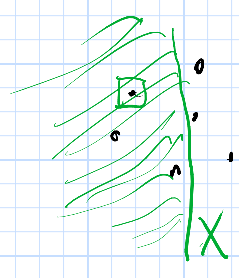
$$= \underline{O(nm)}$$

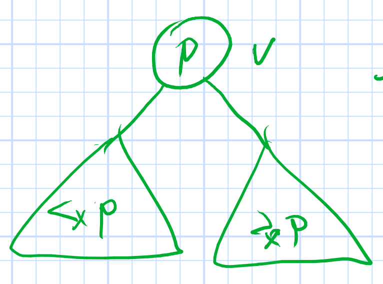
$\leq c\frac{n}{2}m + c'nm \leq cnm.$

Speeding up dp with data structures

<u>Data structure</u>

- insert points (x,y)
- query. given X return point with maximum value of y among all points stored so far.





P_v

$$\boxed{p_v = \arg\max_{(x,y) \in P_v} y}$$

X

$d = depth(T)$

P_u

$O(d)$

<u>Red-black trees</u>

<u>Treap</u>

$O(\log n)$ ins/del/query