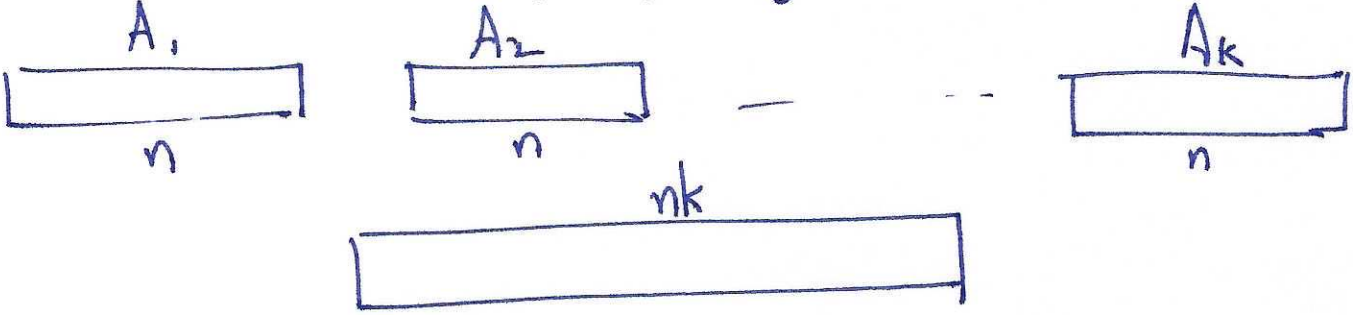
More on design of algorithms and running time analysis

Merge k sorted arrays of length n



B = sort of union of arrays

Trivial solution:

* Copy all arrays into B. Use Mergesort/$O(n\log n)$ sort on B.

~~Running time~~

1) Copy all arrays into output array B. $O(nk)$
2) Sort B using Mergesort. $O(nk\log nk)$

$= O(nk\log n + nk\log k)$

* Repeated merge A_i ~~into~~ with sort of $A_1 \cup A_2 - A_{i-1}$

1) Copy A_1 into B. $O(n)$

2) for $i = 2$ to k

{

(2a) Merge B with A_i into array C. $O(|A_i| + |B|)$ → n, $(i-1)n$

(2b) Copy C to B. $O(|C|) = O(in)$

}

Sorted

by linear time merge

Steps 2a and 2b take $O(in)$ time in i^{th} iteration.

Total running time $= O\left(\sum_{i=2}^{k} in\right) = O(k^2 n) = \boxed{O(nk^2)}$

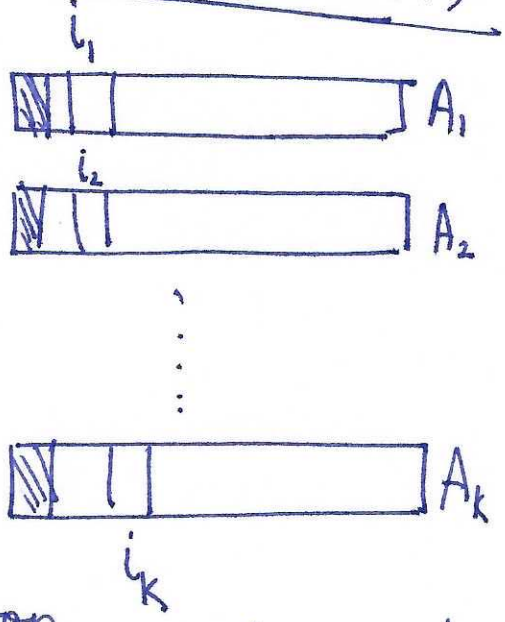
vs $O(nk \log n + nk \log k)$

$\boxed{\sum_{i=2}^{k} i} \leq \boxed{\sum_{i=1}^{k} i} = \frac{k(k+1)}{2}$

$= O(k^2)$

---

*



1) Initialize indices $i_1 = i_2 = \cdots = i_k = 0$ and $i_B = 0$ $O(k)$

2) while $(i_B < nk)$

{

(2a) Find min. among $A_1[i_1], A_2[i_2], \ldots, A_k[i_k]$ using a for loop $O(k)$

(2b) Copy minimum to $B[i_B]$, and increment corresponding (the min.) index i_B++ $O(1)$

(2c) } $O(1)$

Running time analysis: Steps (2a), (2b), (2c) run in $O(k)$ for Step (2). The loop in Step (2) runs nk times, so the total running time is $O(nk^2)$. Step 1 runs in $O(k)$.

So total running time is $O(nk^2 + k) = O(nk^2)$.

$O(2k) = O(k)$

---

1) Create a min-heap H with objects (~~pairs~~ triples) $O(k)$

$O(k)$ $(A_1[0], 0, 1)$ $(A_2[0], 0, 2)$ $(A_3[0], 0, 3)$ — — $(A_k[0], 0, k)$

with first element as key. (val, index in array, ~~index~~ which ~~of~~ array)

$O(1)$ 2) Initialize $i_B = 0$

3) for i_B from 0 to $nk-1$

{

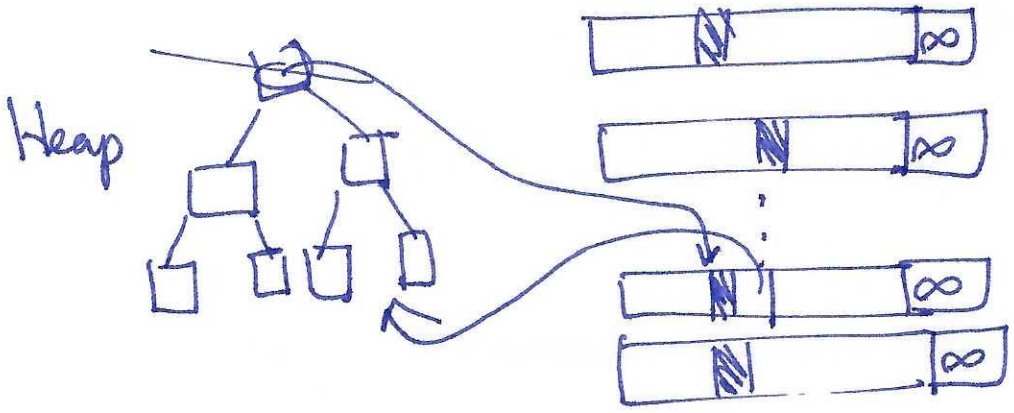
$O(\log k)$ (3a) Extract min from H. Let it be ~~(val, j)~~. (val, ind, j)

$O(1)$ (3b) Set $B[i_B] = val$. Increment i_B.

$O(\log k)$ (3c) Insert $(A_j[ind+1], ind+1, j)$ into H.

}

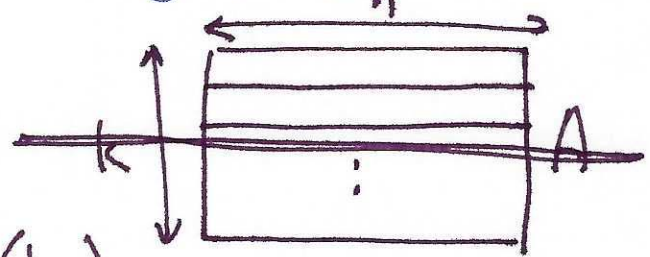
$O(nk \log k)$



Running Time analysis

Step 1 runs in $O(k)$ by BuildHeap procedure.
Step 2 runs in $O(1)$.
Steps (3a) (3c) run in $O(\log k)$ by MinHeap properties
Step (3b) runs in $O(1)$
The loop in Step 3 runs for $O(nk)$ iterations, so the running time is $O(nk \log k)$.
Total running time $= O(k + 1 + nk \log k) = O(nk \log k)$

Let us consider input as a 2D array of size $k \times n$

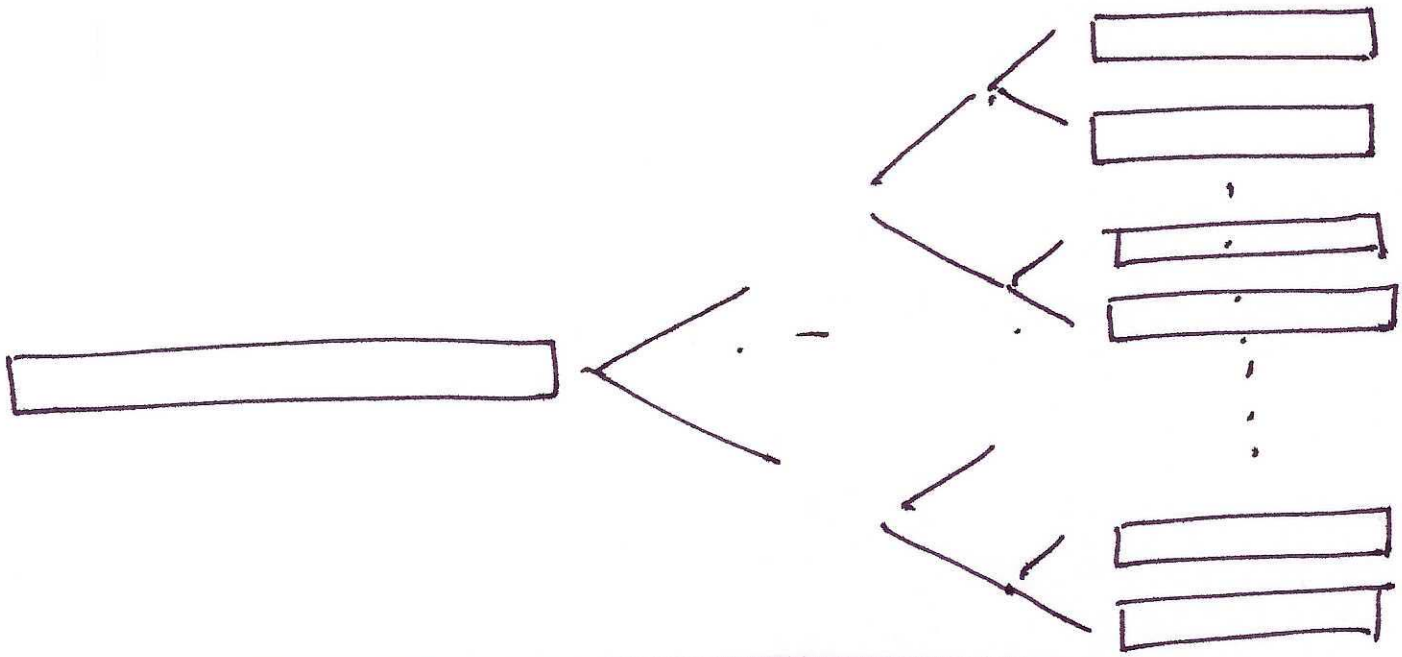


Rows are sorted

0) If $k=1$, return A $\quad O(n)$
1) Split A into A_{up} $\left(\frac{k}{2} \times n\right)$ and A_{down} $\left(\frac{k}{2} \times n\right)$ $\quad O(kn)$
2) Recursively "solve" A_{up} to get array B_{up} $\quad T(k/2)$
3) Recursively "solve" A_{down} to get B_{down}. $\quad T(k/2)$
4) Merge B_{up} ($\frac{kn}{2}$) and B_{down} ($\frac{kn}{2}$) to get output $\quad O(nk)$

$T(k)$ = worst case run time to merge k arrays

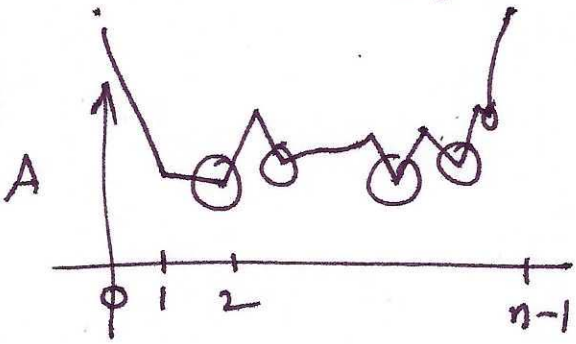
$$T(k) \leq 2T\left(\frac{k}{2}\right) + O(nk) \quad T(k) = O(nk \log k)$$



Q. Find local min of an array. A

i is a <u>local min</u> if $A[i-1] \geq A[i] \leq A[i+1]$ (1)

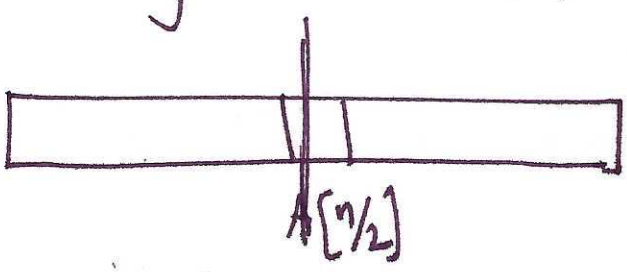
(<u>Assume $A[0] = A[n-1] = \infty$</u>)

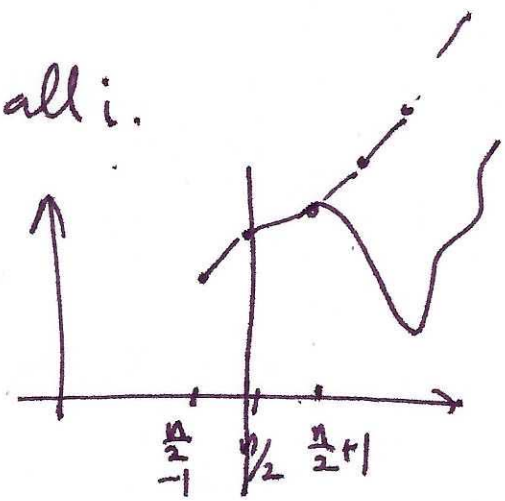


<u>Brute force</u>:

Single for loop checks (1) for all i.

Running time = $O(n)$





<u>Check if $A[n/2]$ is a local min.</u> If it is, done.

If not,

Thm: Consider any portion of the array $A[i..j]$ and suppose $A[i] \geq A[i+1]$ and $A[j] \geq A[j-1]$. Then, there exists a local min. in $A[i..j]$.

localmin(A, start, end) (We guarantee that $A[start] \geq A[start+1]$ and $A[end] \geq A[end-1]$)

0. If $|end - start| \leq 4$, find local min using brute force. $O(1)$

1. Check $A[\frac{start+end}{2}]$ (Compare $A[mid]$) with $A[mid-1]$ and $A[mid+1]$ $O(1)$

2. If $A[mid]$ is local min, done. $O(1)$.

3. If $A[mid] > A[mid-1]$, return localmin(A, start, mid)
   else return localmin(A, mid, end)

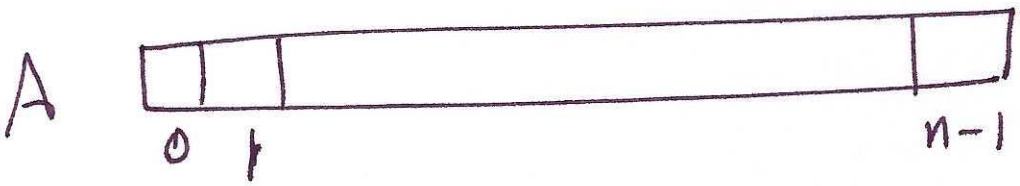
Recurrence is the SAME as binary search

$$T(n) \leq T\left(\frac{n}{2}\right) + O(1)$$

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

Stock Market Problem.

Given an array of prices (each day has a different price), what is the max profit that can be made by one trade?



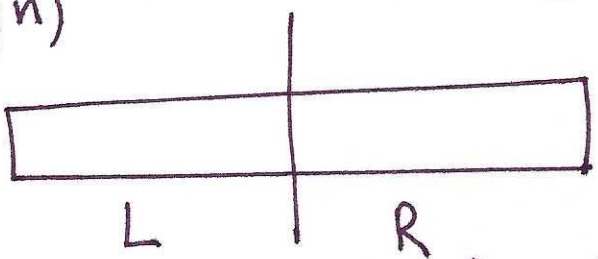
Buy on day i and sell of day $j > i$

$$\text{Profit} = A[j] - A[i]$$

Find the pair of indices $i < j$ s.t $A[j] - A[i]$ is maximized.

Brute force solution gives running time $\Theta(n^2)$.

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



$$T(n) \leq 2T\left(\frac{n}{2}\right) + O(n)$$

Recursively find best profit in L

Recursively find best profit in R

What if you buy in L ($i < \frac{n}{2}$) and sell in R ($j \geq n/2$)?

} Can be solved in $O(n)$