Final Exam: Wed, Dec 15

- comulative to before Fall Break
- No NP-hardness

Flows & Cuts

Given "flow network":

A directed graph $G = (V, E)$, vertices s & t, & capacities $c : E \to \mathbb{R}_{\geq 0}$.

(s, t).

Flow $f : E \to \mathbb{R}_{\geq 0}$: flow in = flow out on all vertices except s & t

flow is feasible if
$f(e) \leq c(e)$

value is $|f|$ = total flow leaving s

Maximum flow: find a feasible (s,t)-flow of max value

An (s,t)-cut (S,T) has $S \subseteq V$, $T \subseteq V$, $S \cap T = \emptyset$,

$S \cup T = V$, $s \in S$

$t \in T$

capacity $\|S,T\|$ = total capacity of edges going from S to T

Minimum cut: find an (s,t)-cut of min capacity

In any flow network with max flow f^* & min cut (S^*, T^*),

$$|f^*| = \|S^*, T^*\|.$$

Ford-Fulkerson: find augmenting paths to push more & more flow

Always $O(E|f^*|)$ if capacities are integers.

$O(E^2V)$ if you choose shortest augmenting paths

Orlin: $O(VE)$ - "theory fast"

- free to cite during exam

applications to bipartite matching, edge-disjoint paths, other "assignment problems"

Divide-and-Conquer

1) Create smaller independent instances of same problem

(mergesort given $A[1..n]$

1) want to sort $A[1..\lfloor n/2 \rfloor]$ + $A[\lfloor n/2 \rfloor+1..n]$

2) Recurse on smaller instances

(mergesort $A[1..\lfloor n/2 \rfloor]$ + mergesort $A[\lfloor n/2 \rfloor+1...n]$

3) Combine results of recursive calls

(merge sorted $A[1..\lfloor n/2 \rfloor]$ + $A[\lfloor n/2 \rfloor+1..n]$)

To analyze: write a run time recurance.

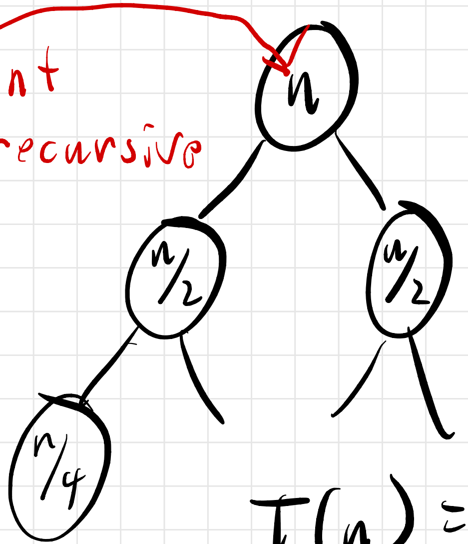
time is $T(n)$

$$T(n) = 2T(n/2) + O(n)$$

mergesort

Solve with recursion trees or Master Method

time spent outside recursive call



$\Sigma = n$

$\Sigma = n$

$\vdots$

$$\underline{T(n) = O(n \log n)}$$

Fall 2019 F P3:

Given array of characters $A[1..n]$.

$\text{IsWord}(i,j)$: True iff $A[i..j]$ is an English word

$\text{NumPartitions}(i)$: # ways to partition $A[i..n]$ into words

$$\text{NumPartitions}(i) = \begin{cases} 1 & i > n \\ \displaystyle\sum_{j=i}^{n} \left([\text{IsWord}(i,j)] \cdot \text{NumPartitions}(j+1) \right) \end{cases}$$

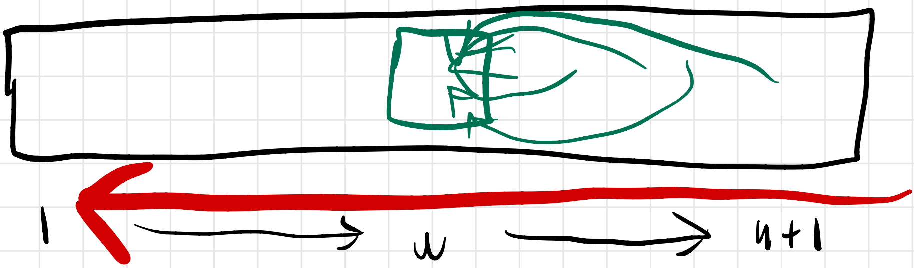
where to end first word

a) memoization DS:

$$1 \le i \le n+1$$

NumPartitions[1..n+1]

b) evaluation order?



right-to-left (for $i \leftarrow n+1$ to 1)

c) space: $O(n)$

time: $O(n) \cdot O(n) = O(n^2)$

time per i

(assuming constant time for IsWord)

d) <u>CountPartitions(A[1..n]):</u>

```
for i ← n+1 down to 1
    if i > n
        NumPartitions[i] ← 1
    else
        total ← 0
        for j ← i to n
            if IsWord(i, j)
                total ← total +
                    NumPartitions
                        [j+1]

return NumPartitions[1]
```

F 2021 M1 P1d

Given $X[1..n]$

$$\text{FindTheValue}(i) = \begin{cases} 0 & \text{if } i > n \\ \max\limits_{i \le j \le n-i+1} \left(j \cdot X[i] + \text{FindTheValue}(i+j) \right) \end{cases}$$

Time to compute FindTheValue(1)?

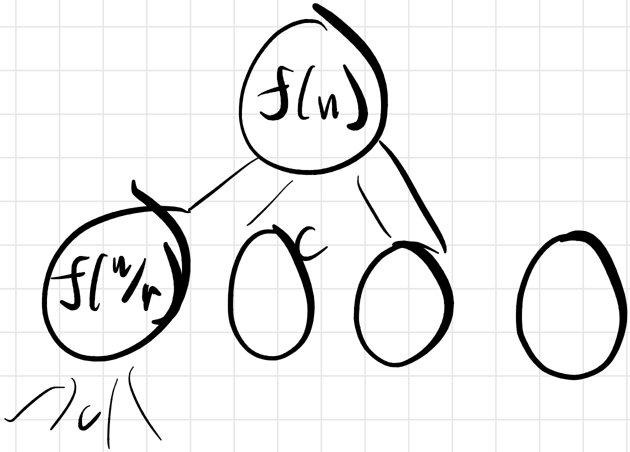
subproblems = $O(n)$

time per subproblem: $O(n)$ (try all j)

total: $O(n) \cdot O(n) = O(n^2)$

Erickson's "Master Method":

$$T(n) = c \cdot T(n/r) + f(n)$$



$f(n)$

$c \cdot f(n/r)$

$c^2 \cdot f(n/r^2)$

Three common cases: (usually if f is a poly)

Level i is $x^i \cdot f(n)$ for $x < 1$

(decreasing geometric level sums)

$$T(n) = \Theta(f(n))$$

Every level = $f(n)$

$$T(n) = \Theta(f(n) \cdot \log n)$$

Level i is $K^i \cdot f(n)$ $(K>1)$

(increasing geometric)

$$T(n) = \Theta(\#\text{leaves})$$

$$= \Theta\left(c^{\log_r n}\right)$$

$$= \Theta\left(n^{\log_r c}\right)$$

F 2017 F P2:

Given a stack of n pancakes.

Can flip/reverse the top k pancakes for any k we choose.

Want to sort from small on top to big on bottom.

Recursion. Can recursively sort top x pancakes with affecting lower ones.

So put biggest on bottom then sort top n-1 pancakes.

SortCakes(n):
  if n=0, return
  Let biggest be k from top.
  Flip(k)
  Flip(n)
  SortCakes(n-1)

\# flips $T(n) = 2 + T(n-1)$
$= 2n$

eval.utdallas.edu