Thursday, April 4, 2024 9:33 AM

Midterm 2 Review:

Divide & Conquer, DP, Graph algo
(BFS, DFS, SCC, Shortest paths ..)

9 A *shuffle* of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of X and Y in the same order. For example, the string $BANANAANANAS$ is a shuffle of the strings $BANANA$ and $ANANAS$ in several different ways.

$\texttt{BANANA}ANANAS \qquad \texttt{BAN}ANA\texttt{ANA}NAS \qquad \texttt{B}AN\texttt{AN}A\texttt{A}NA\texttt{NA}S$

Similarly, the strings $PRODGYRNAMAMMIINCG$ and $DYPRONGARMAMMICING$ are both shuffles of $DYNAMIC$ and $PROGRAMMING$:

$PRO\texttt{D}G\texttt{Y}R\texttt{NAM}AMMI\texttt{I}N\texttt{C}G \qquad \texttt{DY}PRO\texttt{N}G\texttt{A}R\texttt{M}AMM\texttt{IC}ING$

Describe and analyze an efficient algorithm to determine, given three strings $A[1..m]$, $B[1..n]$, and $C[1..m+n]$, whether C is a shuffle of A and B.

Subproblem: $0 \le i \le m$, $0 \le j \le n$.

Is-Shuf(i, j) : True if $C[1 \ldots (i+j)]$ is a shuffle of $A[1 \ldots i]$, $B[1 \ldots j]$

Ans: Is-Shuf(m, n)

Base Case: Is-Shuff$(0,0)$ = True.

$1 \le i \le m$ Is-Shuff$(i, 0)$ = True if $A[1..i] = C[1 \ldots i]$

$1 \le j \le n$ Is-Shuff$(0, j)$ = True if $B[1..j] = C[1..j]$

Formula: $1 \le i \le m$, $1 \le j \le n$

$C[1] \ldots C[i+j]$

$A[1] \ldots A[i]$ $\qquad B[1] \ldots B[j]$

Is-Shuf(i,j) = False if $C[i+j] \ne A[i]$ & $C[i+j] \ne B[j]$

... $A[i]$ & $C[i+j] \ne B[j]$

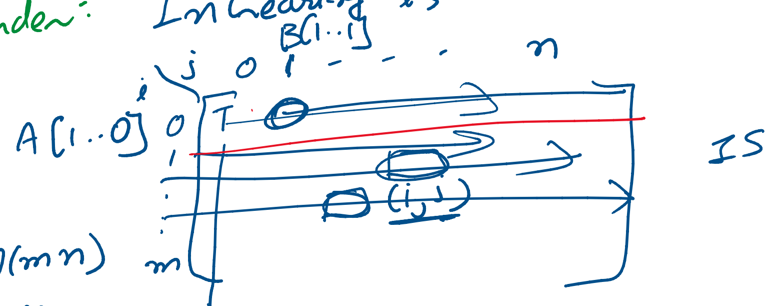
$= \text{Is-Shuf}(i-1, j)$ if $C[i+j] = A[i]$ & $C[i+j] \neq B[j]$

$= \text{Is-Shuf}(i, j-1)$ if " $\neq$ " & " = "

$= (\text{Is-Shuf}(i-1, j))$ if " = " & " = "

OR $(\text{Is-Shuf}(i, j-1))$

Evaluation Order: Increasing i, increasing j.



Running Time:

\# sub prob: $O(mn)$

time/s.p.: $O(1)$

Total: $O(mn)$

Pseudo Code: // $IS[i,j] = \text{Is-Shuf}(i, j) = T/F$

1. $IS[0,0] = T$
2. for $i = 1$ to m do if $(A[1..i] = C[1...i])$ then $IS[i,0] = T$ else $IS[i,0] = F$.
   ~~O(m)~~ $O(m)$
3. for $j = 1$ to n do if $(B[1...j] = C[1..j])$ then $IS[0,j] = T$ else $IS[0,j] = F$.
   ~~O(n)~~ $O(n)$
4. for $i = 1$ to m
5. for $j = 1$ to n
6. if $A[i] = C[i+j]$ & $B[j] = C[i+j]$ then $IS[i,j] = IS[i-1,j] \vee IS[i,j-1]$
7. else if $A[i] = C[i+j]$ & $B[j] \neq C[i+j]$

then IS[i,j] =

7. Else if A[i] = C[i+j] & B[j] ≠ C[i+j]
then IS[i,j] = IS[i-1, j]

8. Else if A[i] ≠ C[i+j] & B[j] = C[i+j]
then IS[i,j] = IS[i, j-1]

9. Else IS[i, j] = False.

10 Suppose you are given a sequence of non-negative integers separated by + and × signs; for example:

$$2 \times 3 + 0 \times 6 \times 1 + 4 \times 2$$

You can change the value of this expression by adding parentheses in different places. For example:

$$2 \times (3 + (0 \times (6 \times (1 + (4 \times 2))))) = 6$$
$$((((2 \times 3) + 0) \times 6) \times 1) + 4) \times 2 = 80$$
$$((2 \times 3) + (0 \times 6)) \times (1 + (4 \times 2)) = 108$$
$$(((2 \times 3) + 0) \times 6) \times ((1 + 4) \times 2) = 360$$

Describe and analyze an algorithm to compute, given a list of integers separated by + and × signs, the smallest possible value we can obtain by inserting parentheses.

Your input is an array $A[0..2n]$ where each $A[i]$ is an integer if i is even and + or × if i is odd. Assume any arithmetic operation in your algorithm takes $O(1)$ time.

Subproblem: $0 \le i \le j \le 2n$ i, j even

Min-Sum(i, j) = Min-sum value for i/p $A[i \ldots j]$

Ans: Min-sum$(0, 2n)$

Base case: Min-Sum(i,i) = $A[i]$ $\forall i$ even

Formula: $0 \le i < j \le 2n$

$A[i\ .]k[\ j]$

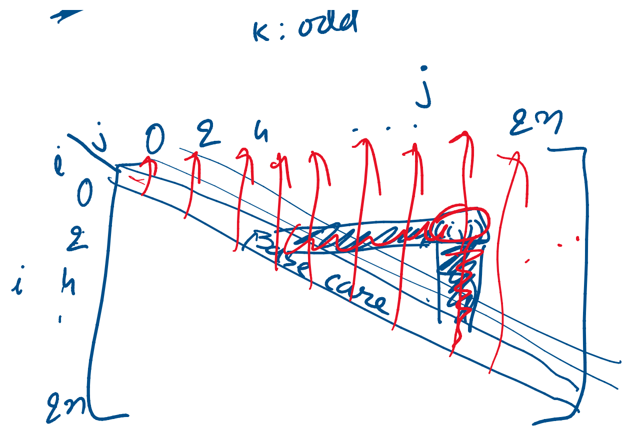
$$\text{Min-Sum}(i,j) = \min_{\substack{i<k<j \\ k:\text{odd}}} \text{Minsum}(i, k-1) \cdot A[k] \cdot \text{Min-Sum}(k+1, j)$$

j

Evaluation:

k : odd

Evalution:



increasing in $(j-i)$
$\equiv$ increasing in j & decreasing in i

29 Suppose you are given a directed graph G in which ***every edge has negative weight***, and a source vertex s. Describe and analyze an efficient algorithm that computes the shortest path distances from s to every other vertex in G. Specifically, for every vertex t:

- If t is not reachable from s, your algorithm should report $dist(t) = \infty$.
- If the shortest-path distance from s to t is not well-defined because of negative cycles, your algorithm should report $dist(t) = -\infty$.
- If neither of the two previous conditions applies, your algorithm should report the correct shortest-path distance from s to t.

(**Hint:** First think about graphs where the first two conditions never happen.)

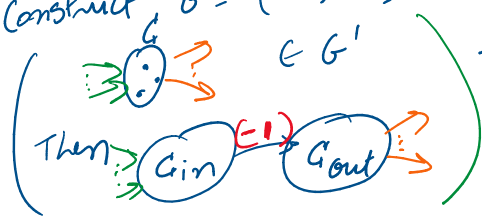
Sol'n sketch:

$O(m+n)$ { (1) Find SCC & their Meta-Graph, say $G' = (V', E')$ (G' is a DAG).

$O(m+n)$ { (2) Do BFS from $scc(s)$ in G'
(3) For each component C_i not in the BFS tree do
for each vertex $v \in C_i$ set $d[v] = \infty$
Remove C_i & all it's incident edges from G'

$O(m+n)$ { (4) Construct $G'' = (V'', E'')$ where $V'' = \{C_{i\,in}, C_{i\,out} \mid C_i \in V'\}$

$C_i \in G'$



& $E'' = \{(C_{i\,out}, D_{in}) \mid (C_i, D) \in E'\}$
$\cup \{(C_{i\,in}, C_{i\,out}) \mid C_i \in V'\}$

Weights on all the edges is 0 except for
$w(C_{i\,in}, C_{i\,out}) = -1$ if #vertices in $C_i > 1$.

(5) G'' is a DAG. Run SSSP on G'' starting at.

... < 0 then

(5) G'' is a DAG. Run SSSP on G'' starting at $SCC(s)_{in}$. For each component C if distance from $SCC(s)_{in}$ to $C_{out} < 0$ then for each $v \in C$ set $d[v] = -\infty$. Remove C & adjacent edges from G'

$O(m+n)$

(6) The remaining components in G' must be singleton vertices. And the remaining graph is still a DAG. So run SSSP starting at s w/ original -ve weights on edges.

$O(m+n)$

Total run time: $O(m+n)$

33 There are n galaxies connected by m intergalactic teleport-ways. Each teleport-way joins two galaxies and can be traversed in both directions. Also, each teleport-way e has an associated cost of $c(e)$ dollars, where $c(e)$ is a positive integer. A teleport-way can be used multiple times, but the toll must be paid every time it is used.

Judy wants to travel from galaxy s to galaxy t, but teleportation is not very pleasant and she would like to minimize the number of times she needs to teleport. However, she wants the total cost to be a multiple of five dollars, because carrying small change is not pleasant either.

33.A. Describe and analyze an algorithm to compute the smallest number of times Judy needs to teleport to travel from galaxy s to galaxy t while the total cost is a multiple of five dollars.

min # edges.

33.B. Solve part (a), but now assume that Judy has a coupon that allows her to use one teleport-way for free.

* Input:
$G = (V, E)$ is the graph of galaxies connected through teleports (TP).
Undirected

V = galaxies
$(u, v) \in E$ if galaxy u is connected to galaxy v via a T.P.

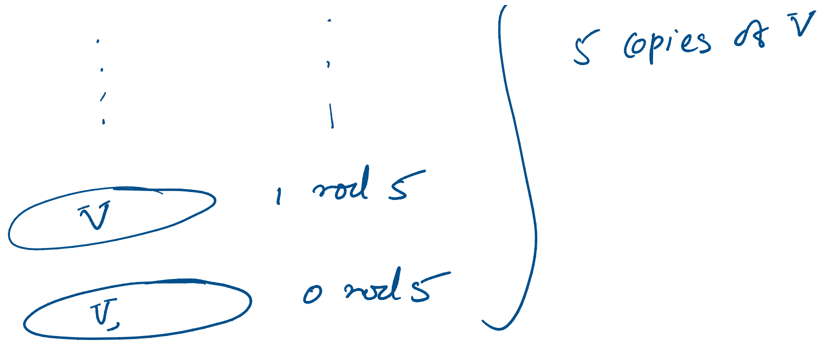
$c(e)$: cost of taking T.P. edge e.
$n = |V|$, $m = |E|$

(33.A): (Idea: keep track of "cost-paid mod 5" in the vertex description of the new graph)

Construct $G' = (V', E')$

V $u \bmod 5$ 5 copies of V

☆ V' :



Formally, $\bar{V} = \{(v,i) \mid v \in V$ & $i \in \{0,1,2,3,4\}\}$

→ $(v,i) \in V'$ represents that Judy is at galaxy $v \in \bar{V}$ & the cost paid so far is i mod 5.

☆ E' :

$u \xrightarrow{c(u,v)} v \in E$

⇓

$(u,i) \longrightarrow (v, (i + c(u,v)) \bmod 5)$

$(v,j) \longrightarrow (u, (j + c(u,v)) \bmod 5)$

→ Formally $E' = \left\{ \left((u,i), (v,k)\right) \;\middle|\; \{u,v\} \in E, \; k = (i + c(u,v)) \bmod 5 \right\}$

$|V'| = 5n = O(n)$ $\quad |E'| = 10m = O(m)$

☆ Find shortest path from $(s,0)$ to $(t,0)$ using BFS in $O(m+n)$ time.

☆ Justification: when Judy starts at vertex s in graph G, she has not paid any cost yet. Hence, the corresponding vertex in G' is $(s,0)$. She

She can reach t in G paying (0 mod 5) cost $\iff$ in G' $\exists$ a path from $(s,0)$ to $(t,0)$

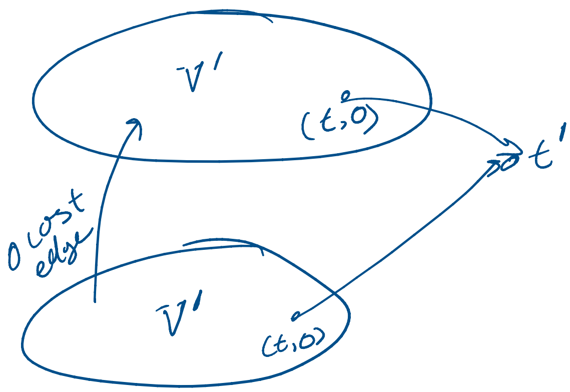
The shortest path will minimize the # edges/teleports cost $\iff$ in G' $\exists$ a path from $(s,0)$ to $(t,0)$

The shortest path will minimize the # edges/teleports taken.

(II. B) (Idea: Make two copies of G')

Construct $G'' = (V'', E'')$

✪ V'':



Formally:

$$V'' = \left\{ (v,i,b) \;\middle|\; \begin{array}{l} (v,i) \in V \;\& \\ b \in \{0,1\} \end{array} \right\} \cup \{t'\}$$

bit b represents # free edges taken.

✪ $$E'' = \left\{ \big((u,i,b), (v,k,b)\big) \;\middle|\; \begin{array}{l} \big((u,i), (v,k)\big) \in E' \\ b \in \{0,1\} \end{array} \right\}$$ (within each of the two copies, keep all the E' edges)

$$\cup \left\{ \big((u,i,0), (v,i,1)\big) \;\middle|\; \{u,v\} \in E \right\}$$ ($(u,i,0) \to (v,i,1)$ represents the free edge)

$$\cup \left\{ \big((t,0,0), t'\big), \big((t,0,1), t'\big) \right\}$$ (both $(t,0,0)$ & $(t,0,1)$ are valid termination points. So connect them to a common terminal t')

$(s, 0, 0)$

terminal t')

★ Find shortest path from $(s, 0, 0)$ to t'.

0 roads cost · no free edge taken so far.

Using again BFS on G' starting at $(s, 0, 0)$

In $O(|E''| + |V''|) = O(2|E'| + |E| + 2, \; 2|V''| + 1)$

$= O(m+n)$ time.

35 After graduating you accept a job with Aerophobes-Я-Us, the leading traveling agency for people who hate to fly. Your job is to build a system to help customers plan airplane trips from one city to another. All of your customers are afraid of flying (and by extension, airports), so any trip you plan needs to be as short as possible. You know all the departure and arrival times of all the flights on the planet.

Suppose one of your customers wants to fly from city X to city Y. Describe an algorithm to find a sequence of flights that minimizes the *total time in transit*—the length of time from the initial departure to the final arrival, including time at intermediate airports waiting for connecting flights.