Thursday, April 7, 2022 10:22 AM

Midterm 2 on April 11. 7-9pm.

Syllabus: Divide & conquere, Dynamic Programming

Graph algo: BFS, DFS, DAG, SCC,

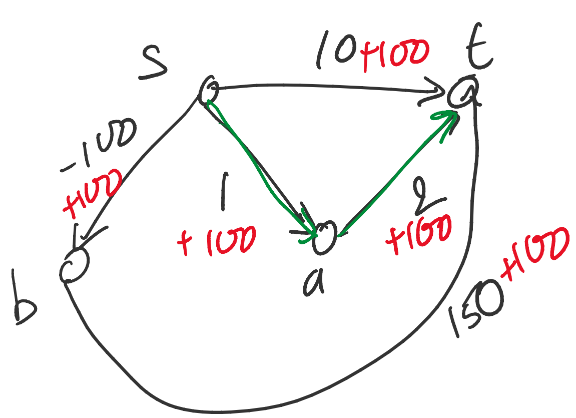
Shortest path algo.

"Please read all the instructions on webpage"

Q: Shortest path problem, can -ve weight be converted to +ve weights?

By $\forall e \in E$ $w(e) \rightarrow H + w(e)$ H: large +ve number.

NO!



S.P.: s→a→t

S.P.: s→t. ✗

10 Suppose you are given a sequence of non-negative integers separated by $+$ and $\times$ 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 $\times$ 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 $\times$ if i is odd. Assume any arithmetic operation in your algorithm takes $O(1)$ time.

☆ Define Subproblem: $i \le j \in [0, \dots, 2n]$, i, j even

$c(i, j)$: is the smallest possible value we can obtain by inserting parentheses to $A[i]\, A[i+1] \dots A[j]$.

☆ Answer: $C(0, 2n)$

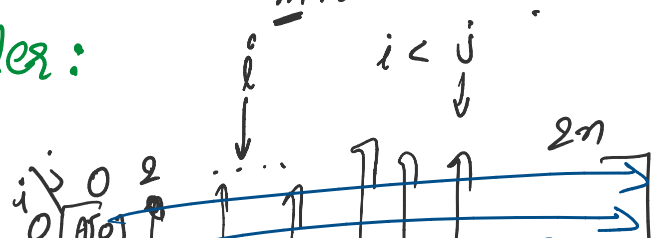
☆ Base case: $c(i, i) = A[i]$, $i \in [0, \dots, 2n]$ i even

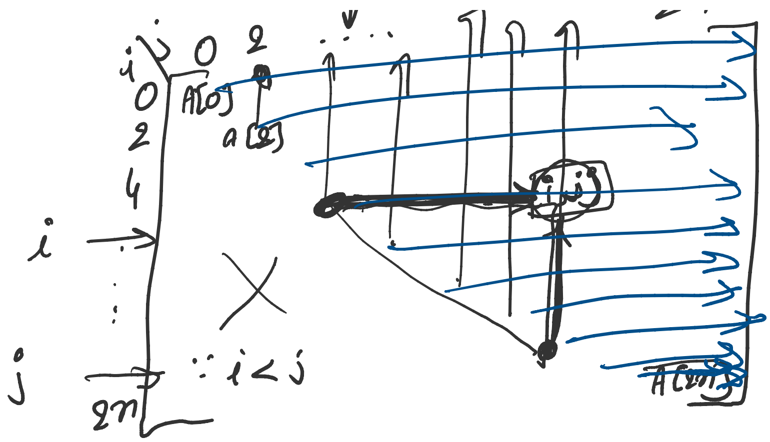
☆ Recursive Formula: $i < j$, i, j are even.

$O(n)$ time.

$$c(i, j) = \min_{\substack{i \le m \le j \\ m:\text{odd}}} c(i, m-1) \cdot \underset{+/\times}{A[m]} \cdot c(m+1, j) \qquad i+2 \le m+1 \le j$$

☆ Evaluation Order: $i < j$ increasing j decreasing i





decreasing i

increasing j

✡ Psuedo code :

```
for i = 0 to 2n            i even
        c[i,i] = A[i]

for j = 0 to 2n            j even
    for i = j+2 to 0       i even
            c[i,j] = ∞
        for m = i+1 to j-1     m odd
            if A[m] = + then
                temp = c[i, m-1] + c[m+1, j]
            Else
                temp = c[i, m-1] x c[m+1, j]
            c[i,j] = min{c[i,j], temp}
        End
    End
End
Output c[0, 2n].
```

✡ Running Time :

$O(n^2)$ sub problems

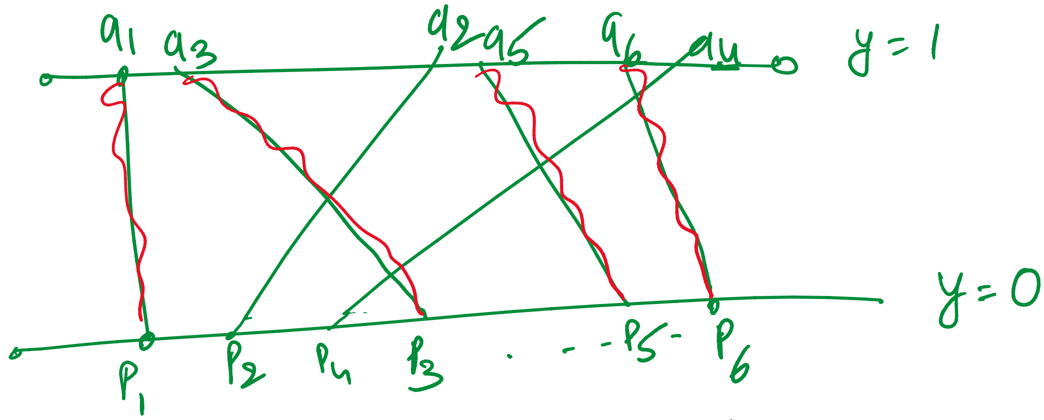
Each takes $O(n)$ time.

Total $O(n^3)$.

Total $O(n^3)$.

13 **13.A.** Suppose we are given a set L of n line segments in the plane, where each segment has one endpoint on the line $y = 0$ and one endpoint on the line $y = 1$, and all $2n$ endpoints are distinct. Describe and analyze an algorithm to compute the largest subset of L in which no pair of segments intersects.

13.B. Suppose we are given a set L of n line segments in the plane, where each segment has one endpoint on the line $y = 0$ and one endpoint on the line $y = 1$, and all $2n$ endpoints are distinct. Describe and analyze an algorithm to compute the largest subset of L in which ***every*** pair of segments intersects.



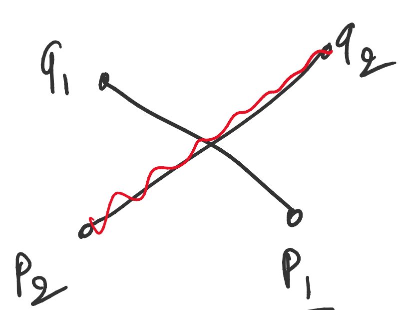
13.A

Given a set of line seg. $L = \{(p_1, q_1), \ldots, (p_n, q_n)\}$

Find largest subset $S \subseteq L$ s.t. no two seg in S intersect.

Assume: segments are ordered s.t. $q_1 < q_2 < \ldots < q_n$

Intersection / crossing:



Intuition:

$q_1 < q_2$ & $p_1 > p_2$

$\iff$ 1 & 2 intersects

If we take (p_2, q_2)

P_2 P_1

If we take (P_2, q_2) then any seg that is taken prior to it should have $\boxed{p\text{-cond} < P_2}$

Sol'n: Similar to longest increasing subsequence.
(LCS of $P_1 P_2 \dots P_n$)

13. B

Observation: In the above figure if we take (P_1, q_1) then we can take (P_2, q_2) iff $P_2 < P_1$

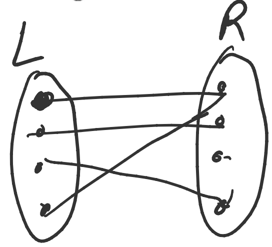
$\Downarrow$

Sol'n: Similar to longest decreasing subsequence
(LDS of $P_1 P_2 \dots P_n$, assuming $q_1 < q_2 < \dots < q_n$)

24 A graph (V, E) is bipartite if the vertices V can be partitioned into two subsets L and R, such that every edge has one vertex in L and the other in R.

24.A. Prove that every tree is a bipartite graph.

24.B. Describe and analyze an efficient algorithm that determines whether a given undirected graph is bipartite.



Bipartite. $\Rightarrow$ No odd cycles.

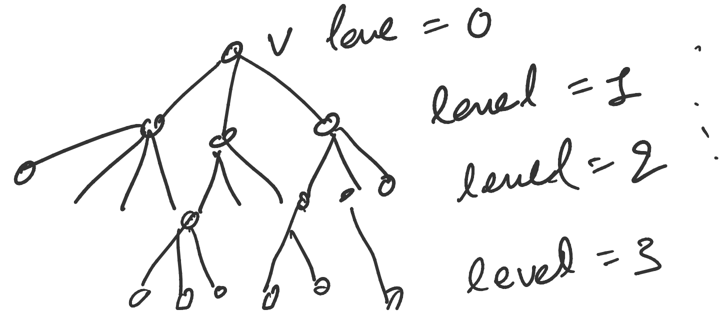
(A): $\rightarrow$ Root the tree at $v \in V$

$\forall u \in V$, $\exists$ a unique path from v to u.

path length from v to u.

$\forall u \in V$, $\exists$ a unique path ...... r to u.

$\text{leve}(u)$ = path length from r to u.



$L = \{u \mid \text{level}(u) = \text{even}\}$

$R = V \setminus L. = \{u \mid \text{level}(u) = \text{odd}\}$

Claim: $(u,v) \in E$ then either $u \in L$ & $v \in R$
OR $u \in R$ & $v \in L$.

Pf: Let $\text{level}(u) = i$ then $\text{level}(v) = i+1$ or $i-1$

case I: $\text{level}(v) = i+1$

If i is odd then $i+1$ is even

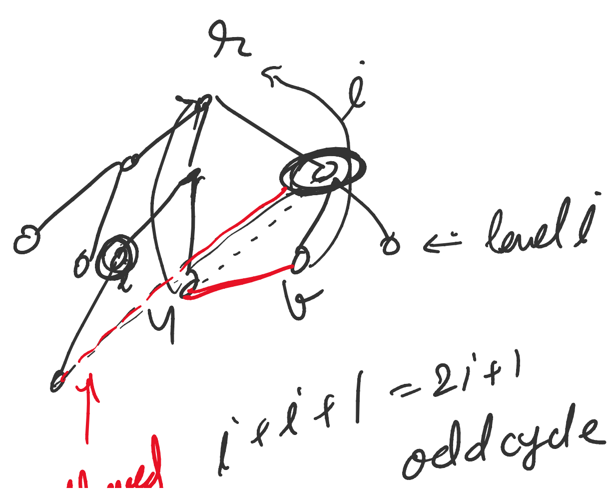
$\Rightarrow u \in R$ & $v \in L$

o.w. $u \in L$ & $v \in R$.

case II: symetric.

(3) 1. Do BFS.

2. Check no cross edges in the same level.



same level.

Not allowed ∴ BFS.

odd cycle

Running time:

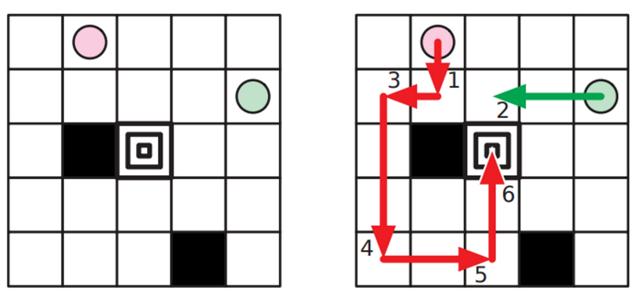
line 1 : $O(m+n)$

line 2 : $O(m)$

Total : $O(m+n)$

26 ***Kaniel Dane*** is a solitaire puzzle played with two tokens on an $n \times n$ square grid. Some squares of the grid are marked as *obstacles*, and one grid square is marked as the *target*. In each turn, the player must move one of the tokens from is current position *as far as possible* upward, downward, right, or left, stopping just before the token hits (1) the edge of the board, (2) an obstacle square, or (3) the other token. The goal is to move either of the tokens onto the target square.

For example, in the instance below, we move the red token down until it hits the obstacle, then move the green token left until it hits the red token, and then move the red token left, down, right, and up. In the last move, the red token stops at the target *because* the green token is on the next square above.



An instance of the Kaniel Dane puzzle that can be solved in six moves.
Circles indicate the initial token positions; black squares are obstacles; the center square is the target.

Describe and analyze an algorithm to determine whether an instance of this puzzle is solvable. Your input consist of the integer n, a list of obstacle locations, the target location, and the initial locations of the tokens. The output of your algorithm is a single boolean: TRUE if the given puzzle is solvable and FALSE otherwise. The running time of your algorithm should be a small polynomial in n.

Construct Directed Graph $G = (V, E)$

$V = \{((i,j), (i',j')) \mid (i,j) \neq (i',j')$ & $(i,j), (i',j') \neq$ obstacles $\} \cup \{t\}$

$V = \left\{ ((i,j), (i',j')) \;\middle|\; \begin{array}{l} (i,j), (i',j') \neq \text{obstacles} \\ i,j,i',j' \in [0,\dots,n] \end{array} \right\} \cup \{t\}$

$|V| = O(n^4)$

$E = ((i,j), (i',j')) \longrightarrow ((i \pm \delta_1, j \pm \delta_2), (i' \pm \delta_3, j' \pm \delta_4))$

exactly one of $\delta_1, \delta_2, \delta_3, \delta_4$ are non-zero.

δ_i s.t. no obstacle/ boundary /oken token is hit.

$\boxed{\text{out-deg}(v) \leq 8}$ $\forall v \in V \Rightarrow |E| = O(n^4)$

$((t_1,t_2), *) \longrightarrow t$

$(*, (t_1,t_2)) \longrightarrow t$

start position $s = (s_1, s_2)$

Q: can I reach t from s? $\Rightarrow$ BFS.

Running time: $O(n^4)$