Midterm 2 on April 11, 7-9pm.

Syllabus: Divide & Conquer, Dynamic Programming

Graph algo: BFS, DFS, DAG, SCC, Shortest path algo.

"Please read all the instructions on webpage!"

Q: Shortest path problem, can we weight be converted to the weights?

By $w(e) \rightarrow H + w(e)$ $H$: large +ve number

No!

![Diagram of a graph with nodes S, b, a, d, g, and edges with weights]

s.p.: s→a→t
s.p.: s→t. x
Suppose you are given a sequence of non-negative integers separated by + and \times signs; for example:

\[
(2 \times 3) + \left(0 \times (6 \times (1 + (4 \times 2)))\right)
\]

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:** \(i \leq j \leq [0, \ldots, 2n] \), \(i, j\) even

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

**Answer:** \(c(0, 2n)\)

**Base Case:** \(c(i, i) = A[i]\), \(i \in [0, \ldots, 2n]\), \(i\) even

**Recursive Formula:** \(i < j\), \(i, j\) are even

\[
c(i, j) = \min_{m \in \{i \ldots j\}} \left\{ c(i, m-1) \times A[m] \times c(m+1, j) \right\}
\]

**Evaluation Order:**

- \(i < j\)
- Increasing \(j\)
- Decreasing \(i\)
Pseudo code:

1. for $i = 0$ to $2n$ do even
2. \[ c[i, i] = A[i] \]
3. for $j = 0$ to $2n$ do even
4. for $i' = j+2$ to $0$ do even
5. for $m = i+1$ to $j-1$ do odd
6. if $A[m] = +$ then
7.   \[ \text{temp} = c[i, m-1] + c[m+1, j] \]
8. else
9.   \[ \text{temp} = c[i, m-1] \times c[m+1, j] \]
10. end if
11. \[ c[i, j] = \min \{ c[i, j], \text{temp} \} \]
12. end for
13. end for
14. Output $c[0, 2n]$.

Running Time: $O(n^2)$ subproblems
Each takes $O(n)$ time.
Total $O(n^3)$. 
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.
If we take \((P_2, q_2)\)
then any seq that is taken prior to it should have
\[ p_{\text{rand}} < p_2 \]

Solu'n: Similar to longest increasing subsequence.

\((\text{LCS of } p_1, p_2, \ldots, p_n)\)

13. B

Observation: In the above figure if we take \((p_1, q_1)\)
then we can take \((P_2, q_2)\) if \( p_2 < p_1 \)

\[ \frac{\downarrow}{\text{Solu'n: Similar to longest decreasing subsequence}} \]

\((\text{LDS of } p_1, p_2, \ldots, p_n, \text{assuming } q_1 < q_2 < \cdots < q_m)\)

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.

\[ \text{Bipartite} \Rightarrow \text{No odd cycles.} \]

\( (A) : \Rightarrow \text{Root the tree at } v \in V \)

\[ \forall u, v \in V, \text{ a unique path from } u \text{ to } v. \]

... path length from \(v\) to \(u\).
Claim: $(u, v) \in E$ then either $u \in L$ & $v \in R$ or $u \in R$ & $v \in L$.

Proof:

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: symmetric.

(3) 1. Do BFS.

2. Check no cross edges in the same level.
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 its 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 $n$. 

\[
\begin{align*}
\text{Construct } & \quad \text{Graph} \\
G = (V, E) \\
V = \{((i,j), (i',j')) | \text{ obstacles} \} \cup \{(i,j), (i',j') \}
\end{align*}
\]
\[ V = \left\{ (i, j), (i', j') \mid (i, j), (i', j') \neq \text{obstacles} \right\} \cup \{ \text{obstacles} \} \]

\[ |V| = O(n^4) \]

\[ E = ( (i, j), (i', j') ) \rightarrow \left( (i \pm \delta_1, j \pm \delta_2), (i' \pm \delta_3, j' \pm \delta_4) \right) \]

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

subject to no obstacle/boundary token hit.

\[ \text{put-deg} (v) \leq 8 \quad \forall v \in V \Rightarrow |E| = O(n^4) \]

\[ ((t_1, t_2), \#) \rightarrow t \]

\[ (\#, (t_1, t_2)) \rightarrow t \]

Start position \( s = (s_1, s_2) \)

Q: Can I reach \( t \) from \( s \)? \( \Rightarrow \) BFS.

Running time: \( O(n^4) \)