

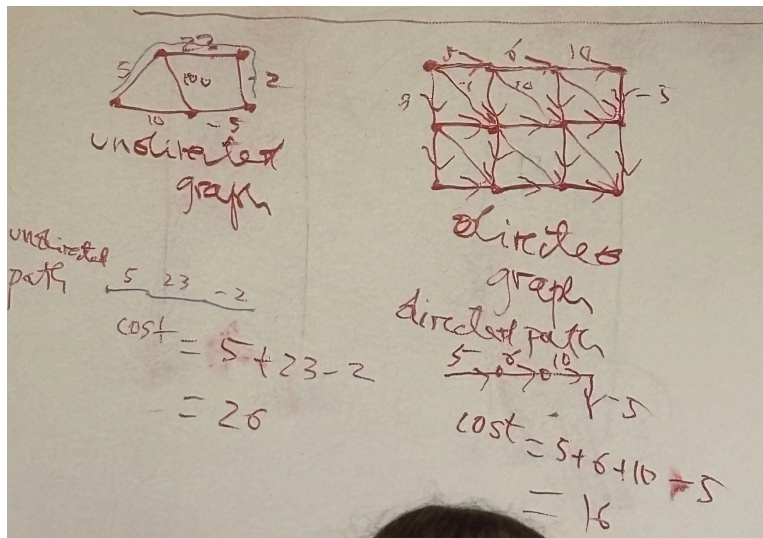
10/8/26

Temple of sequence alignment

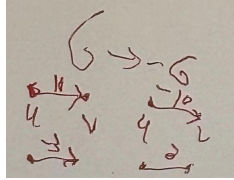
- CS
- Bio
- Probability & Statistics

Longest path problems and shortest paths problems w/ graphs

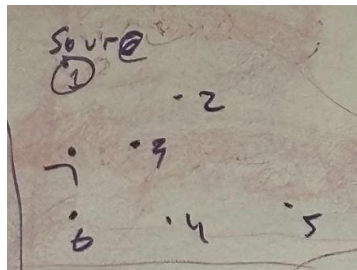
- Undirected graphs
 - Typically numbers on the edges that represent the cost of the edge
 - Easier to deal with
- Directed graphs
 - E.g. edit graphs
 - Each edge has a direction
 - Can also have costs with numbers



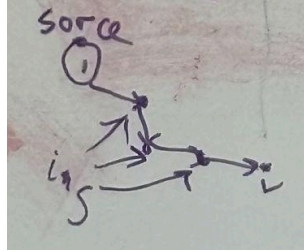
- Longest paths
 - Max cost paths
 - Alignment $\longleftrightarrow$ directed path
 - Max score alignment $\longleftrightarrow$ longest cost path
- Shortest Path PB (problem) $G = (V, E)$
 - Solvable in polynomial time
 - Dijkstra's algorithm
 - $O(|V|^2)$
- DAG (Directed Acyclic Graphs)
 - $O(|V| + |E|)$
 - Linear time
 - Allows us to optimally solve the shortest path and the longest path with the same algorithm



-
- The longest path PB is NP complete
 - Computational intractable
 - Famous problem is P vs NP
 - Universally believe $P \neq NP$
- Professor Istrail recounted his visit with Professor Dijkstra
- Dijkstra's shortest path algorithm
 - $G = (N, A)$ directed graph
 - N = set of vertices
 - A = set of edges
 - Consider node and vertex equivalent
 - Each edge has nonnegative cost/length
 - One node is the source vertex
 - PROBLEM:
 - Given: $G = (N, A)$, $N = \{1, 2, \dots, n\}$, A edges (i, j) $L[i, j]$ cost of the edge (i, j)
 - Compute: The cost of the shortest paths from the source vertex to each of the other nodes in G and shortest paths
 - Two sets C, S of vertices
 - S = set of vertices already chosen
 - S contains all the vertices whose minimal path cost from the source is already known
 - C = set of candidate vertices
 - C = the rest of the vertices
 - At each step, we choose a node in C whose cost of the path to the source is the smallest and add it to S



-
- Def: A path from the source node to a node v is called special if all intersection nodes along the path are in S
- When we add a new node from C to S then the shortest special path to v is the shortest path in G



-
- The algorithm ends with all nodes in S,
 - D gives us the solution: shortest paths costs
 - $N = \{1, 2, \dots, n\}$
 - $L[i, j] \geq a$ if edge (i, j) exists in A
 - Cost of the edge (i, j)
 - $L[i, j] = \text{infinity}$ if (i, j) is not an element of A

Dijkstra's Algorithm

{initialization}

$S = \{1\}$ # 1 = source node

$C = \{2, 3, \dots, n\}$

FOR $i = 2$ to n DO

$D[i] = L[1, i]$

{GREEDY LOOP}

Repeat $n-2$ times

$V = \text{node in } C \text{ with minimum } D[v]$

$S = S \cup \{v\}$

$C = C \setminus \{v\}$

FOR $w \in C$ DO

IF $D[w] > D[v] + L[v, w]$

THEN

$D[w] = D[v] + L[v, w]$