Tutorial Session

February 15, 2023

Graph Algorithms

- Basic concepts:
  1. graph $G = (V, E)$ (directed/undirected);
  2. presentation of graph: matrix/adjacent list.

Graph Algorithms

- Basic concepts:
  1. graph $G = (V, E)$ (directed/undirected);
  2. presentation of graph: matrix/adjacent list.
- DFS/BFS:
  1. algorithm and running time; (important)
  2. DFS tree and its properties: classification of edges, discovery time and finishing time.

Graph Algorithms

- Basic concepts:
  1. graph $G = (V, E)$ (directed/undirected);
  2. presentation of graph: matrix/adjacent list.
- DFS/BFS:
  1. algorithm and running time; (important)
  2. DFS tree and its properties: classification of edges, discovery time and finishing time.
- Application of DFS/BFS (important):
  1. single-site shortest path in unit graph: BFS;
  2. connected components: BFS/DFS;
  3. topological sort (definition and algorithm);
  4. strongly connected component (definition, Korusaju's algorithm and Tarjan's algorithm).

Graph Algorithms

- Basic concepts:
  1. graph $G = (V, E)$ (directed/undirected);
  2. presentation of graph: matrix/adjacent list.
- DFS/BFS:
  1. algorithm and running time; (important)
  2. DFS tree and its properties: classification of edges, discovery time and finishing time.
- Application of DFS/BFS (important):
  1. single-site shortest path in unit graph: BFS;
  2. connected components: BFS/DFS;
  3. topological sort (definition and algorithm);
  4. strongly connected component (definition, Korusaju's algorithm and Tarjan's algorithm).
- Hint: you may use these algorithms as black-boxes in exam so long as the problem does not require you to describe them in detail.

Selection of problems in PS (PS9-5)

- Given a tree $G = (V, E)$ rooted at $r \in V$, answer the following type of query in $O(1)$ time within $O(n)$ preprocessing time.
  - is u an ancestor of v?

Selection of problems in PS (PS9-5)

- Given a tree $G = (V, E)$ rooted at $r \in V$, answer the following type of query in $O(1)$ time within $O(n)$ preprocessing time.
  - is u an ancestor of v?
- u is an ancestor of v iff $[e_v, f_v] \subseteq [e_u, f_u]$.

Selection of problems in PS (PS9-6)

- Given a DAG $G = (V, E)$, you may perform the following operation:
  - remove a subset of vertices $S \subseteq V$ such that the indegree of any vertex $v \in S$ is 0.
- Find out the minimum number of operations to remove all vertices in linear time.

Selection of problems in PS (PS9-6)

- Given a DAG $G = (V, E)$, you may perform the following operation:
  - remove a subset of vertices $S \subseteq V$ such that the indegree of any vertex $v \in S$ is 0.
- Find out the minimum number of operations to remove all vertices in linear time.
- Topological sort + dynamic programming.
- f_u be the minimum number of operations to remove vertex u.
- $f_u = \max_{v \in N(u)} f_v + 1$.

Selection of problems in PS (PS9-7)

- Snakes and Ladders.
- Use BFS to solve this problem.
- dynamic programming is not a valid solution (partial order of states is required for dynamic programming).

Selection of problems in PS (PS11-6)

- Binary search the answer and run BFS starting from s.

Selection of problems in PS (PS10-1)

- Determine if there exists a vertex $s \in V$ such that s is reachable from all other vertices.

Selection of problems in PS (PS10-1)

- Determine if there exists a vertex $s \in V$ such that s is reachable from all other vertices.
- If such s exists, s must be in the sink SCC and any vertex in sink SCC satisfies.
- Use Tarjan's SCC algorithm, find out any arbitrary vertex v in sink SCC and verify if v satisfies the given property.
- You may write such solution in exam.

Selection of problems in PS (PS10-2)

- Given a graph $G = (V, E)$, determine if G is weak connected.

Selection of problems in PS (PS10-2)

- Given a graph $G = (V, E)$, determine if G is weak connected.
- Without loss of generality, we assume that G is a DAG.
- Suppose $v_1, v_2, \ldots, v_n$ is a topological order of G.
- G is weak connected iff $(v_i, v_{i+1}) \in E$.

Greedy

- minimum spanning tree (definition, algorithms and some proofs) (important), huffman coding and etc.
- approximate algorithms (may appear in exam)
- There exists no approach to determine wheter a problem can be solved via greedy and please be much careful to use greedy without clear evidence or proof of correctness.

Selection of problems in PS (PS10-5)

- Given a graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}_{>0}$ and MST $T = (V, E')$.

Selection of problems in PS (PS10-5)

- Given a graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}_{>0}$ and MST $T = (V, E')$.
- update the weight of $e \in E'$ with weight $w' > w_e$: remove e in MST and add edge $e' = (u, v)$ with minimum weight that connects two components of MST.

Selection of problems in PS (PS11-2)

- minimize average completion time.

Selection of problems in PS (PS11-2)

- minimize average completion time.
- Without release time: sort according to the duration of tasks;
- With release time: pick the task with least duration time upon releasing or finishing a task. (proof sketch: consider the lower bound of c_i.)

Selection of problems in PS (PS11 bonus)

- Algorithm for set cover: until $\mathcal{S}$ covers U, add set S_i to $\mathcal{S}$ with most cost-effectiveness.
- Analysis and tightness: http://www14.in.tum.de/personen/khan/Arindam

Shortest path

- Dijkstra's algorithm, Bellman-ford's algorithm, Floyd's algorithm (algorithm, time complexity and constraints on input).
- detect negative cycle in graph.
- longest path in DAG (can be treated as an algorithm based on DP).

Selection of problems in PS (PS12-1)

- Let f_u be the earliest starting time of job u: $f_u = \max_{v \in N_+(u)} f_v + w_u$ (i.e., the longest path from t to u).
- Let g_u be the latest starting time of job u without affecting project's duration: $g_u = \min_{v \in N_-(u)} g_v - w_u$ (i.e., f_t minus the longest path from t to u).

Selection of problems in PS (PS12-5)

- add $e = (x, y)$ in transition closure $G = (V, E)$: add all $(u, v) \in V^2$ into E such that $(u, v) \notin E, (u, x) \in E$ and $(y, v) \in E$.
- optimization: if $(u, y) \in E$ or $(x, v) \in E$, such edges are already in E.
- time complexity analysis: if $(u, x) \in E$ but $(u, y) \notin E$ before this update, (u, y) will be added into E.

dynamic programming

- knapsack problem, LIS/LCS, maximum independent set of tree, subset sum and etc.
- ordering of states
    - total ordering: $\{1, 2, \ldots, n\}$;
    - partial ordering: dp on DAG, segment dp, subset dp, dp on tree, etc.
- techniques for designing states of dynamic programming.

Selection of problems in PS (PS13-2)

- $f_{i,j}$: maximum profit for cutting a rod of length j with maximum length i.
- time complexity: $O(n^2 \log n)$.

Selection of problems in PS (PS13-3)

- rooted the tree arbitrarily.
- Let $f_{u,0/1}$ be the minimum size of set cover of subtree rooted at u such that u is occupied/not occupied

Selection of problems in PS (PS13-5)

- $dp_{i,j}$ be the optimal score the first player will obtain if i cards from the left and j cards from the right have been taken.
- fun fact: provided that the sum of values are odd, the first player will always win the game. (why?)

computability and complexity

- Turing machine, halting problem, P versus NP, NPC.
- classical problems in NPC: 3-SAT, hamiltonian path, subset sum, knapsack problem (why is this in NPC instead of P?)
- reduction.