Final Review Sheet 660

Instructor: Hoa Vu

December 2019

1 Topics before the midterm

- Master theorem
- Mergesort, Heapsort
- Dynamic programming (knapsack, subset sum, longest increasing subsequence).

2 Depth First Search

You need to remember the structure of DFS:

```
Function DFS(G)
    clock = 1
    Initialize visited[v] = false for all vertices v
    for each vertex v = 1, 2 ..., n do
        if visited[v] = false then
            Explore(v)
```

```
1 Function Explore(v)
2     visited[v] = true
3     pre[v] = clock
4     clock = clock + 1
5     for each edge vu (or v → u in directed graph) do
6         if visited[u] = false then
7             Explore(u)
8     post[v] = clock
9     clock = clock+1
```

Algorithm 1: For step 5: assume we go through u from smaller IDs to larger IDs

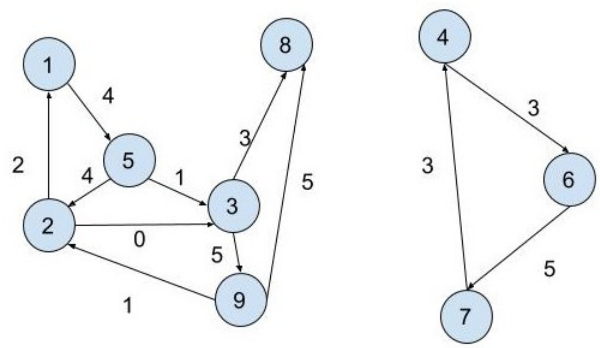
Question 1. State the following definitions:

- Tree edges:
- Forward edges:
- Back edges:
- Cross edges:

Question 2.

- If $v \to u$ is a tree/forward edge, then what is the relationship among $pre[v], post[v], pre[u], post[u]$?
- If $v \to u$ is a cross edge, then what is the relationship among $pre[v], post[v], pre[u], post[u]$?
- If $v \to u$ is a back edge, then what is the relationship among $pre[v], post[v], pre[u], post[u]$?

Question 3. Classify the edges in the below graph. List the pre and post numbers of all the vertices.



Question 3. What is the running time of DFS?

Question 4. A graph is acyclic if and only if . . . .

Question 4. Describe the algorithm that decides if the graph is a acyclic?

Question 5. Topological sort is to order the vertices from left to right such that . . . .

Question 6. To find a topological ordering, we first perform a DFS on the graph and then sort the vertices according to . . . .

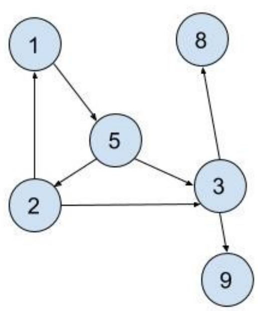
3 Breadth First Search

To find shortest paths from s to every other vertex v on an unweighted graph we use BFS.

1 **Function** *BFS(G,s)*
2   For all $u \in V$, set $dist(u) = \infty$
3     $dist(s) = 0$.
4     $Q = \{s\}$ is a queue.
5   **while** *Q is not empty* **do**
6         $u = eject(Q)$.
7   **for** *each edge $uv \in E$ (or $u \to v$ in directed graphs)* **do**
8   **if** $dist(v) = \infty$ **then**
9                 $push(Q, v)$
10                 $dist(v) = dist(u) + 1$

Algorithm 2: For step 7: assume the loop goes through u from smaller IDs to larger IDs

Question 1. Let s be the vertex 1 in the below graph. List the vertices that we eject from the queue over time.



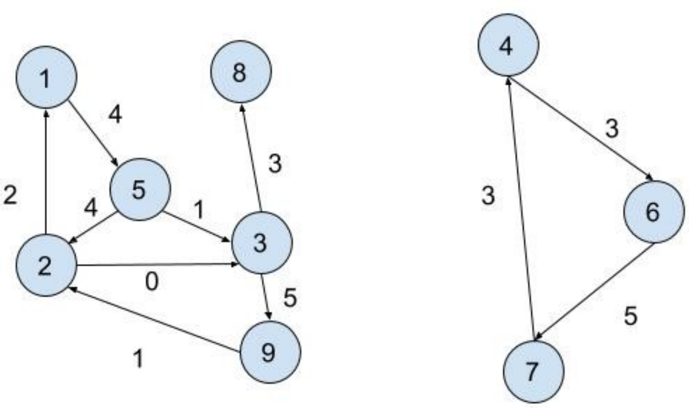
Question 2. What are the shortest paths' distance of each vertex from s found by BFS?

Question 3. What is the running time of BFS?

4 Dijkstra's algorithm

Function $Dijkstra(G, s)$
    For all $u \in V \setminus \{s\}$, initialize $dist(u) = \infty$.
    $dist(s) = 0$.
    while $R \neq V$ **do**
        Pick $u \notin R$ with smallest $dist(u)$, then $R \leftarrow R \cup \{u\}$.
        for *each vertices* $uv \in E$ *(or* $u \rightarrow v$ *in directed graphs)* **do**
            if $dist(v) > dist(u) + \ell(uv)$ **then**
                $dist(v) = dist(u) + \ell(uv)$.

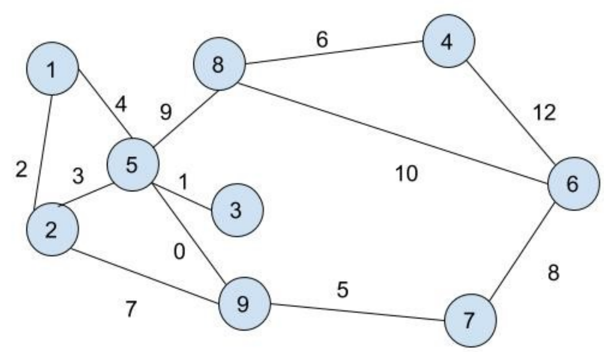
Algorithm 3:



Question 1. We run Dijsktra algorithm with the source vertex 1. For each vertex, list their distinct *dist* values over time.

Question 2. What is the running time of the above version of Dijstra algorithm?

5 Floyd-Warshall algorithm



Question 1. Floyd-Warshall algorithm is to find all-pair shortest paths, TRUE or FALSE?

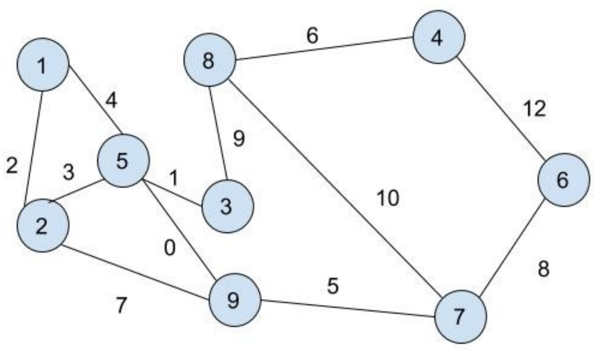
Question 2. In the dynamic programming, what is the formula for the base case $D[i, j, 0]$?

Question 3. In the dynamic programming, what is the formula for the base case $D[i, j, k]$?

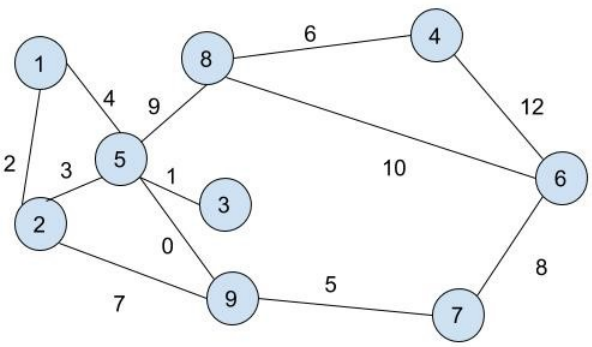
Question 4. Ihe above graph, what is $D[2, 7, 4]$? What is $D[1, 7, 5]$?

Question 5. What is the running time of Floyd-Warshall?

6 Minimum Spanning Tree



Question 1. Draw the MST of the above graph using Kruskal's algorithm. Label the edges in the order that the algorithm picks.



Question 2. Draw the MST of the above graph using Prim's algorithm. Label the vertices and edges in the order that the algorithm picks.

7 NP-Completeness

Question 1. $P \subseteq NP$, TRUE or FALSE?

Question 2. State the definition of (decision) Problem A reduces to (decision) Problem B.

Question 3. State the definition of an NP problem:

Question 4. State the definition of an NP-Complete problem:

Question 5. Let X and Y be two decision problems. Suppose we know that X reduces to Y. Which of the following can we infer?

- If Y is NP-complete then so is X.
- If X is NP-complete then so is Y.
- If Y is NP-complete and X is in NP then X is NP-complete.
- If X is NP-complete and Y is in NP then Y is NP-complete.
- X and Y can't both be NP-complete.
- If X is in P, then Y is in P.
- If Y is in P, then X is in P.

Question 6. If A is NP-Complete and $P \neq NP$, then there may still be a polynomial time algorithm for A? TRUE or FALSE?

Question 7. SET-PARTITION: Given a set S of n integers. Decide whether S can be partitioned into S_1 and S_2 (i.e., $S_1 \cup S_2 = S$ and S_1, S_2 are disjoint) such that $\sum_{x \in S_1} x = \sum_{y \in S_2} y$. Show that SET-PARTITION is NP-Complete (hint: consider a reduction from SUBSET-SUM).

8 Maximum Flow

Question 1. Draw the residual network of the following flow.

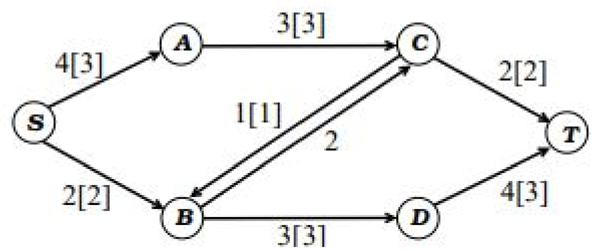


Figure 16.2: A network flow graph with positive flow shown using "capacity[flow]" notation.

Question 2. Is the above a max flow? If so, why? If not, get a larger flow.

Properties of network flow Make sure you know the properties of network flow and its application to maximum bipartite matching.

9 Approximation algorithms.

Algorithms from lectures Know the approximation algorithms for vertex cover, metric TSP, maximum set coverage.