

Q1

15 Points

Consider the following problem on a dictionary of n words, $W_1 \dots W_n$, each with exactly k characters.

You can transform a word W_i into word W_j if they differ in at most $d \leq k$ characters. (both d and k are specified as part of the input, along with n and the words)

For example, if the dictionary is:

$W_1 = \text{'hit'}$, $W_2 = \text{'cog'}$, $W_3 = \text{'hot'}$, $W_4 = \text{'dot'}$, $W_5 = \text{'dog'}$, $W_6 = \text{'lot'}$, $W_7 = \text{'log'}$, and $d = 1$, one way to change 'hit' to 'cog' is:

'hit' \rightarrow 'hot' \rightarrow 'dot' \rightarrow 'dog' \rightarrow 'cog'.

We want to find the fewest number of steps to transform W_1 to W_2 .

Q1.1

5 Points

I claim that this problem can be formulated as a shortest path problem. Please provide a strategy to formulate this as a shortest path problem and give a graph visualization of the problem

From the question, we can create a vertex V_i for all of the words W_i .

Then, we're going to make an edge from i to j if W_i can be changed into W_j by changing at most d characters and use BFS to find an efficient path from the start to the end word. And the graph is an undirected graph and the edges are unweighted.

Please see the attached file for the graph visualization.

▼ S-Q1.1.pdf

Download

1 / 1

-

+

**Q1.2**

5 Points

Show that your graph (in Q 1.1) can be constructed in $O(n^2)$ time, and its size is up to $O(n^2)$.

Note: We definitely can consider the graph construction as being $O(n^2k)$ by comparing every pair of words.

From the question, we can know that the graph can be constructed in $O(n^2)$ time complexity at worst case. So, we can

think all the words W_1, W_2, \dots, W_n as the vertices of the graph. So the graph will be directed.

Then, we will create n vertices for all the n words after we have defined the graph and we will iterate through the list of words n times for each word and to check the other words that can be converted to. When we found a word in the list and we can add an edge between those two vertices and which will take a nested loop and a outer loop until we reach to n and so we can say that the time complexity is $O(n^2)$. In the best case, it still be $O(n^2)$.

So, we can think the size of the graph is $O(n^2)$ and we can use BFS to traverse the graph and find the shortest path between those two words.

 No files uploaded

Q1.3

5 Points

Using your formulation above, give a $O(n^2k + n^3)$ time algorithm for finding the minimum number of steps (you only need to describe how to find the number of steps, not the sequence of transformations).

We gonna use BFS to find the minimum steps.

First, we can use a queue and to keep the record of the depth where we traverse the point and we also initialize a set that has been visited and to keep a record of which vertices are visited. Next, we started to expand the path for one for each time and start from the path from the first of our queue and we can go further one more step from our last visited vertex.

Then, we can explore our queue continuously when we pop out a path from our queue and we then retrieve the last vertex visited and its neighbor vertexes.

Last, We can remove visited vertexes. Then we recursively add the vertex to visited and add a path that is constructed of the path until now and adding the vertex for each of the remaining neighbor vertexes which are unvisited.

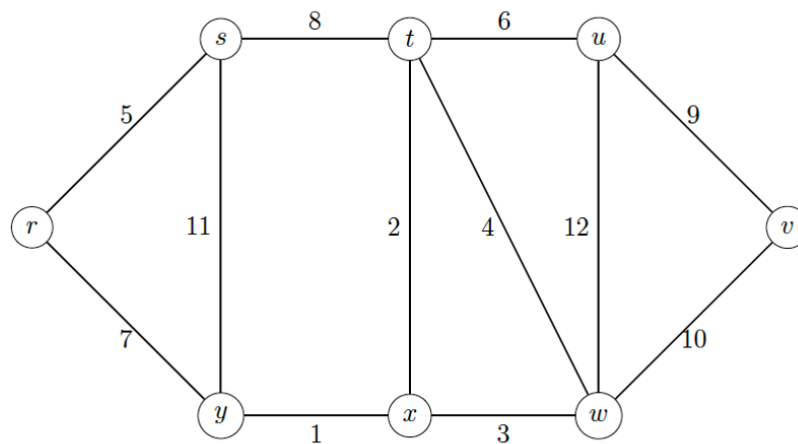
So, we can do a $O(n^2k + n^3)$ time algorithm for finding the minimum number of steps.

 No files uploaded

Q2

5 Points

Consider an undirected graph on 8 vertices, with 12 edges given as shown below:



Q2.1

2 Points

Give the result of running Kruskal's algorithm on this edge sequence (specify the order in which the edges are selected).

(y, x) -> (x, t) -> (x, w) -> (r, s) -> (t, u) -> (r, y) -> (u, v)

 No files uploaded

Q2.2

3 Points

For the same graph, exhibit a that certifies that the edge ry is in the minimum spanning tree. Your answer should be in the form of $E(S, V \setminus S)$ for some vertex set S . Specifically, you should find S .

Because the edge (s, y) is the edge cut not taken, so we can know that edge (r, y) is in the MST.

From the question, we can know that the cut vertices: $\{r, y\}$ and this cut divide the graph G into two parts: $S1: \{r, s\}$ and $S2: \{t, v, w, x, y, z\}$

Then, we know the cut set of the graph G is $\{st, sy, ry\}$

So, the minimum weighted edge from the cut set should be list in the MST of graph G according to the cut property.

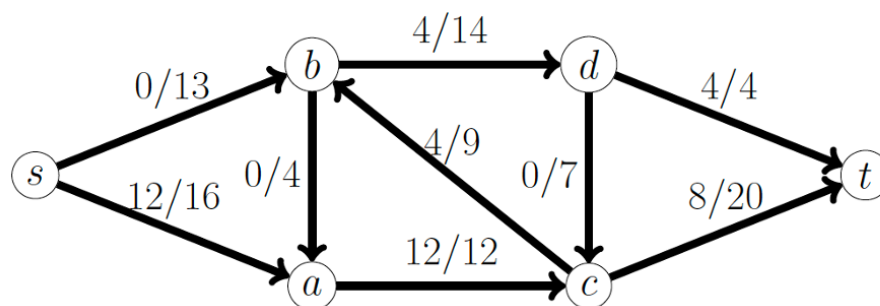
So, we get the minimum from the cut set which is $ry = 7$ and the ry is also in the MST.

 No files uploaded

Q3

20 Points

Consider the following directed network with flows written as the first number and edge capacity as the second on each edge:



Q3.1

5 Points

Draw the residual network obtained from this flow.

Please see my attached file.

▼ S-Q3.1.pdf

 Download

1 / 1

-

+

**Q3.2**

10 Points

Perform two steps of the Ford Fulkerson algorithm on this network, each using the residual graph of the cumulative flow, and the augmenting paths and flow amounts specified below. After each augment, draw two graphs, preferably side by side; these are graphs of:

- The flow values on the edges
- Residual network

The augmenting paths and flow amounts are:

i) $s \rightarrow b \rightarrow d \rightarrow c \rightarrow t$ with flow amount **7** Units

Please see my attached file.

▼ S-Q3.2.pdf

Download



ii) $s \rightarrow b \rightarrow c \rightarrow t$ with **4** units.

Note for continuity your second graph should be coming from the one in (i) NOT from the initial graph.

Please see my attached file.

▼ S-Q3.2(ii).pdf

Download



Q3.3

5 Points


Exhibit a maximum flow with flow values on the edges, state its value, and exhibit a cut (specified as a set of vertices) with the same

value.

From the attached graph, we can get the max flow is 23.

▼ S-Q3.3.pdf

Download



Q4
10 Points

Recall the **Clique problem**: given a graph G and a value k , check whether G has a set S of k vertices that's a clique. A clique is a subset of vertices S such that for all $u, v \in S$, uv is an edge of G .

The goal of this problem is to establish the NP-hardness of Clique by reducing **VertexCover**, which is itself an NP-hard problem, to Clique. Recall that a vertex cover is a set of vertices S such that every edge uv has at least one endpoint (u or v) in S , and the VertexCover problem is given a graph H and a value l , check whether H has a vertex cover of size at most l .

Note that all these problems are already phrased as decision problems, and you only need to show the NP-Hardness of Clique. In other words, we will only solve the reduction part in this problem, and you DO NOT need to show that Clique is in NP.

Q4.1

5 Points

Let S be a subset of vertices in G , and let C be the complement graph of G (where uv is an edge in C if and only if uv is not an edge in G).

Prove that for any subset of vertices S , S is a vertex cover in G if and only if $V \setminus S$ is a clique in C .

Note: this is an if and only if proof, i.e. you need to show both directions for full credit.

We know that from the question, S is a set of vertices that contains one endpoint of every edge of the graph at least if S is a vertex cover.

So, for any edges will be covered in S and there will be no any edges in vertices of $V \setminus S$ and all vertices in $V \setminus S$ will be connected. Because we know that uv is an edge in C iff which is not an edge in G and thus all those vertices in $V \setminus S$ can be in C .

So, we can think that $V \setminus S$ is a clique in C .

We also know from the question is that if $V \setminus S$ is a clique in C and thus all vertices in $V \setminus S$ can be connected. So, we will have one endpoint in S at lease for any given other edges.

So, we can conclude that for any subset of vertices S is a vertex cover in G .

 No files uploaded

Q4.2

5 Points

Part 4.1 implies the following result (which you may use without proof): G has a vertex cover of size at most k if and only if the complement of G has a clique of size at least $n - k$.

Use this fact to give a reduction from `VertexCover` to `Clique`.
Your solution should have the following two steps:

i) First, show the reduction: specify how the inputs to `VertexCover`, G and k , can be transformed to a valid input pair, H and l , for `Clique`. Make sure to explain why this takes polynomial time.

ii) Second, show that the answer to $\text{Clique}(H, l)$ can be converted to the answer of $\text{VertexCover}(G, k)$. One possibility is to explain how a YES answer to $\text{Clique}(H, l)$ must also mean YES to $\text{VertexCover}(G, k)$, AND a NO answer to $\text{VertexCover}(G, k)$ must also mean NO to $\text{Clique}(H, l)$.

(Hint: The claim we proved in part a) is an if and only if statement.)

(i)

From the question, we can know that a given graph G and integer k and we want to know if G has vertex cover which size is k or not.

To get this, we gonna create a graph H and integer l and also graph H is the complement graph of G and the integer l equals tot the integer m and m is the number of vertices of G .

So, we can get the $\text{Clique}(H, l)$.

For the operation that to generate the complement graph that can be done in a polynomial time because that we just need to traverse over all pairs of vertices in the graph at one time.

So, it is a polynomial time.

(ii)

From the question, we know that for any given graph G has vertex cover in size k and S is the vertex cover set, so we can

know that all edges in graph G must be incident of set S and there will not be any edges in those vertices in $V \setminus S$.

So, we can consider Yes to $\text{Clique}(H, l)$ also mean Yes to $\text{VertexCover}(G, k)$.

But, for another scenario, if S is not in a VertexCover in G , then we should know there will be other edges existing and for the graph H and $V \setminus S$ will not be a clique. As we know the v in vertices and edges can't be connected.

So, we can conclude that G has a vertex cover of size at most k if and only if the complement of G has a clique of size at least $n - k$.

 No files uploaded

Synthesis 2

● GRADED

STUDENT

Kejian Tong

TOTAL POINTS

49 / 50 pts

QUESTION 1

(no title)

14 / 15 pts

1.1 (no title)

5 / 5 pts

1.2 (no title)

5 / 5 pts

1.3 (no title)

4 / 5 pts

QUESTION 2

(no title)

5 / 5 pts

2.1 (no title)

2 / 2 pts

2.2 (no title)

3 / 3 pts

QUESTION 3

(no title)

20 / 20 pts

3.1 (no title)

5 / 5 pts

3.2	(no title)	10 / 10 pts
3.3	(no title)	5 / 5 pts
QUESTION 4		
	(no title)	10 / 10 pts
4.1	(no title)	5 / 5 pts
4.2	(no title)	5 / 5 pts