CS 240 Homework 2

▯▯▯

TOTAL POINTS

6 / 6

QUESTION 1

1 1 / 1

✓ **- 0 pts** Correct

- 0.2 pts partly correct

- 0.4 pts more than 2 graph wrong

- 0.6 pts more than 4 graph wrong

- 1 pts totally wrong

QUESTION 2

2 1 / 1

✓ **- 0 pts** Correct

- 1 pts There is no guarantee that one person in each department will be selected

- 1 pts Why separate staff from his corresponding class? How can you ensure that the selected staff matches its class?

- 0.5 pts Direct all edges from D to C with capacity 1? What if there are no ClassA employees in d1?

- 1 pts Direct all edges from D to C with capacity 1? What if there are no ClassA employees in d1? The Supply can not be considered as a source node.

- 1 pts Pi means department? Then What if there are no ClassA employees in Pi?

- 0.3 pts How can a staff connect to multiple classes

- 1 pts Department can not belongs to X-class

- 1 pts Need to combine the class node, and set appropriate weight

- 0.1 pts Since no separate staff node is set, the result needs to be selected from the staff of the same class in the same department

- 1 pts The capacity from s to Si should be 1.

- 1 pts What if there are no ClassA employees in di?

- 1 pts One staff can not belong to multiple departments.

- 0.5 pts What if there are no ClassA employees in di?

- 0.5 pts There must be m1 number of A-class staff members, m2 number of B-class staff members, m3 number of C-class staff members, and m4 number of D-class staff members

- 0.2 pts staff should only link to corresponding department and class

- 0.2 pts t node is undefined

- 0.5 pts s and t node are undefined

- 0.5 pts capacities are undefined

- 0.1 pts Details are unclear

- 1 pts Click here to replace this description.

- 0.2 pts The capacity of source to class node is undefined

- 0.5 pts Details are unclear

- 1 pts Try maximum flow

QUESTION 3

3 1 / 1

✓ **- 0 pts** Correct

- 0.4 pts Wrong graph

- 0.4 pts Wrong algorithm descriptions

- 0.1 pts Wrong result

- 1 pts totally wrong

- 0.2 pts Inaccuracy of description

QUESTION 4

4 1 / 1

✓ **- 0 pts** Correct

- 0.5 pts After iterating, no volume changes? Conflict with point 2

- 1 pts Click here to replace this description.

- 0.1 pts True and False are reversed

- 0.3 pts ▯▯▯▯▯▯▯▯

- 0.3 pts Lack of detail

QUESTION 5

5 1 / 1

✓ **- 0 pts** Correct

- 0.8 pts Wrong

- 0.5 pts Wrong detail

- 1 pts No answer

- 0.2 pts Unclear description

QUESTION 6

6 1 / 1

✓ **- 0 pts** Correct

- 0.2 pts Incorrect DP

- 0.2 pts didn't split node, w(li,ri)=1

- 0.2 pts Incorrect node connection.

w(ri,lj)=1 if j<i, x_j<x_i, dp(j)=dp(i)+1

- 0.2 pts Incorrect source to node.

Only connect the node whose value in DP is one.

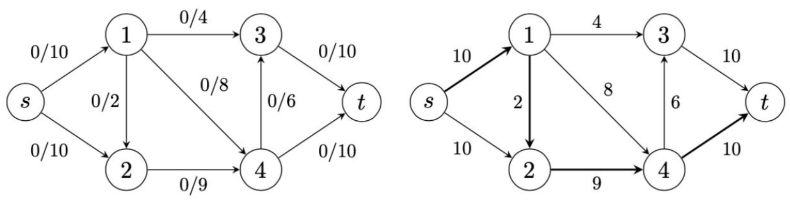
- 0.2 pts Incorrect sink to node.

Only connect the node whose value in DP is L.

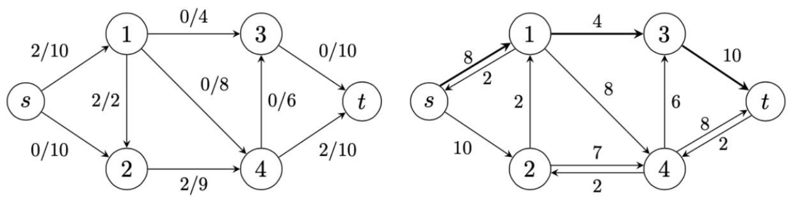
- 0.8 pts Incorrect count subsequence

Problem 1

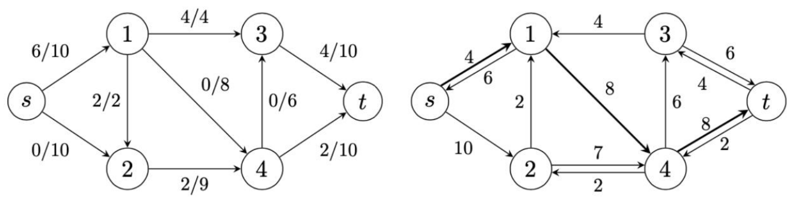
Solution
The process of running the algorithm can be shown as the following figures. The left hand side of the figures are the original graph, and the right hand side is the residual network.
The initial graph:



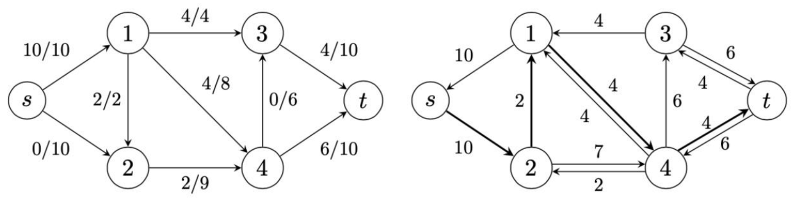
After augmenting 2 by $s \to 1 \to 2 \to 4 \to t$:



After augmenting 4 by $s \to 1 \to 3 \to t$:

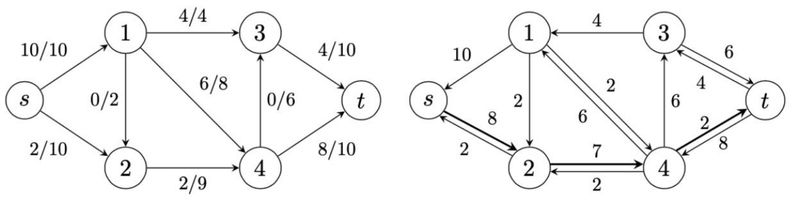


After augmenting 4 by $s \to 1 \to 4 \to t$:

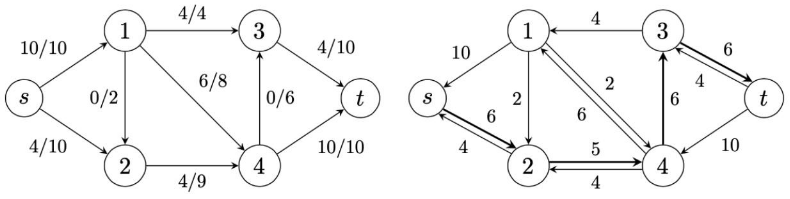


After augmenting 2 by $s \to 2 \to 1 \to 4 \to t$:

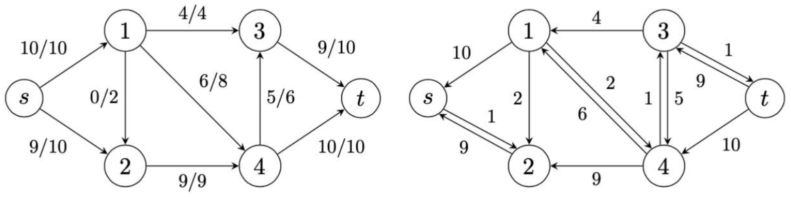
Problem 1 continued on next page. . .



After augmenting 2 by $s \to 2 \to 4 \to t$:



After augmenting 5 by $s \to 2 \to 4 \to 3 \to t$:



Now there is no augmenting path in the residual network so that the algorithm terminates with the max-flow of capacity 19.

Problem 2

Solution

We can construct a graph by following steps:

1. Add a source s, and connects it to k departments $d_1, \dots, d_k$, each edge has *capacity* $= 1$.
2. Connect each department with staff corresponding to it. (e.g. If s_1, s_3 and s_5 belongs to d_1, then add edge $d_1 \to s_1$, $d_1 \to s_3$, $d_1 \to s_5$). Each edge *capacity* $= 1$.
3. Connect each staff to the class he/she belongs to. (e.g. If s_1 belongs to class A, then add edge $s_1 \to A$). Each edge has *capacity* $= 1$.
4. Connect four classed A,B,C and D to t, with capacity m_1, m_2, m_3, m_4.

The graph is shown in the following figure:

1 1 / 1

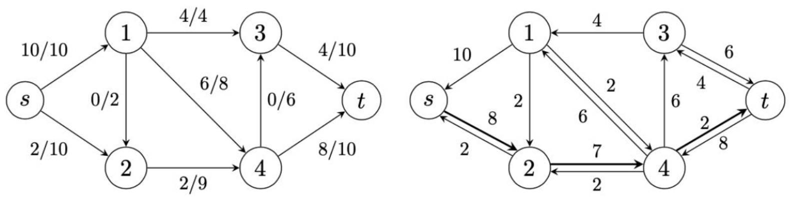
✓ **- 0 pts** Correct

- 0.2 pts partly correct

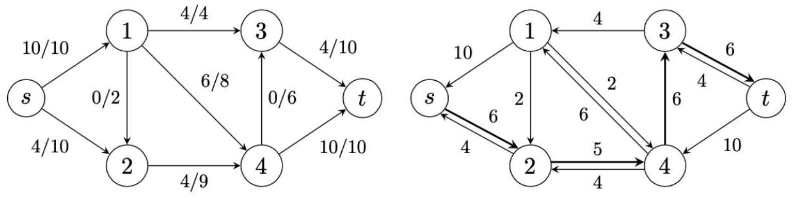
- 0.4 pts more than 2 graph wrong

- 0.6 pts more than 4 graph wrong

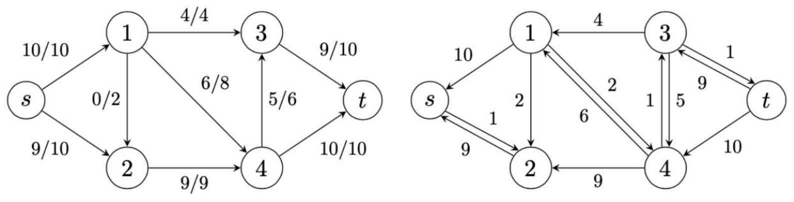
- 1 pts totally wrong



After augmenting 2 by $s \to 2 \to 4 \to t$:



After augmenting 5 by $s \to 2 \to 4 \to 3 \to t$:



Now there is no augmenting path in the residual network so that the algorithm terminates with the max-flow of capacity 19.

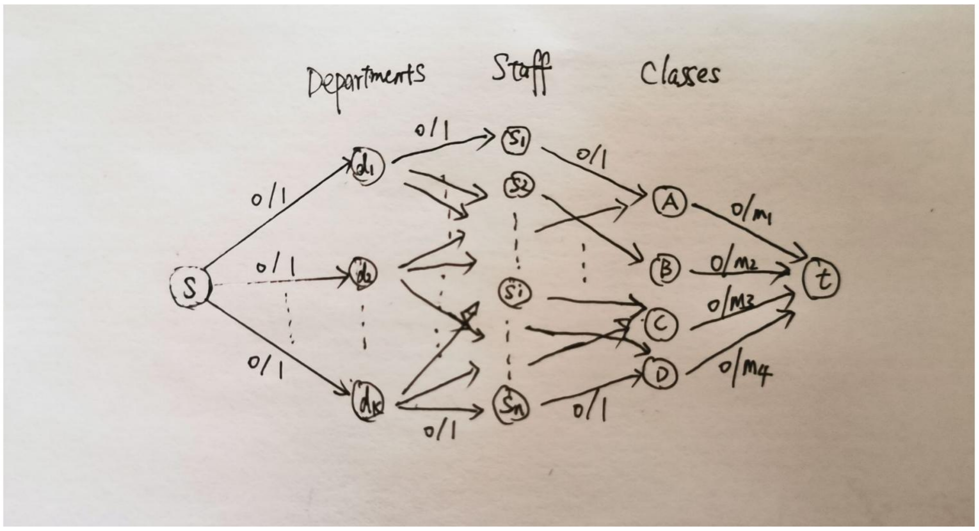
Problem 2

Solution

We can construct a graph by following steps:

1. Add a source s, and connects it to k departments $d_1, \ldots, d_k$, each edge has *capacity* $= 1$.
2. Connect each department with staff corresponding to it. (e.g. If s_1, s_3 and s_5 belongs to d_1, then add edge $d_1 \to s_1$, $d_1 \to s_3$, $d_1 \to s_5$). Each edge *capacity* $= 1$.
3. Connect each staff to the class he/she belongs to. (e.g. If s_1 belongs to class A, then add edge $s_1 \to A$). Each edge has *capacity* $= 1$.
4. Connect four classed A,B,C and D to t, with capacity m_1, m_2, m_3, m_4.

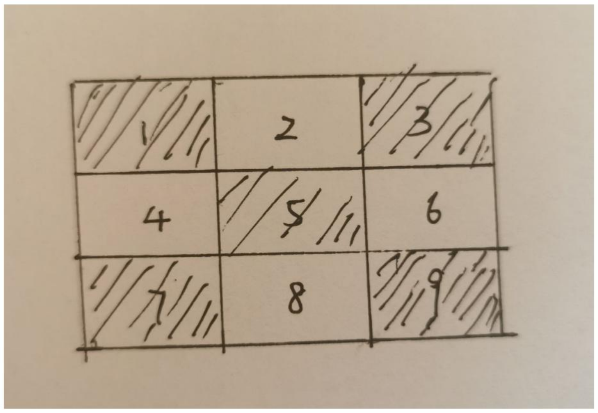
The graph is shown in the following figure:



Then, run Ford-Fulkerson algorithm on this graph, then, if $departmentOf(s_i) \rightarrow s_i \rightarrow classOf(s_i)$ has $flow = 1$ in the max-flow, then s_i is selected. Hence, we have determined who should be selected, in $O(VE^2)$ time (BFS based), where $V = k + n + 6$, for k departments, n staff, 4 classes, s and t, and $E = k + 2n + 4$ for n edges representing $s \rightarrow d_i$, n edges representing $d_i \rightarrow s_j$, n edges representing $s_j \rightarrow class$, and 4 edges representing $class \rightarrow t$.

Problem 3

Solution
First, we divide the matrix into 2 parts, the first part contains $(2i-1, 2j-1)$ and $(2i, 2j)$, the other part contains $(2i-1, 2j)$ and $(2i, 2j-1)$, like the white grids and black grids in chessboard (shown in the following figure).



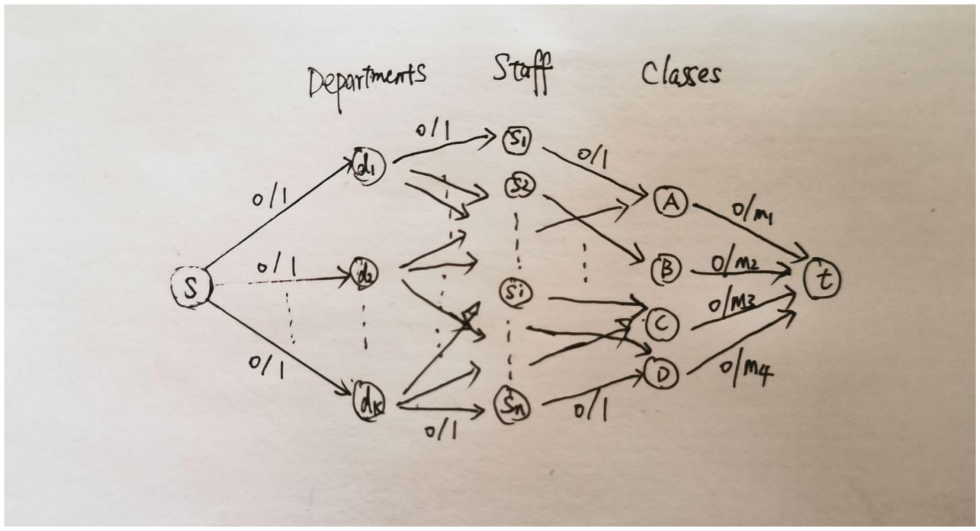
Then, we can construct a graph by following steps:

1. Add a source s connects to all part-I (black) points (1,3,5,7,9 in this figure), with capacity equals to the corresponding value in matrix($cap(s \rightarrow P_{ij}) = Mat[i, j]$).
2. Each part-I points connects to all neighboring part-II (white) points (i.e. 1 connects to 2 and 4, 5 connects to 2,4,6,8), with $capacity = INF$.

Problem 3 continued on next page. . .

2 1 / 1

- ✓ **- 0 pts** Correct
- **- 1 pts** There is no guarantee that one person in each department will be selected
- **- 1 pts** Why separate staff from his corresponding class? How can you ensure that the selected staff matches its class?
- **- 0.5 pts** Direct all edges from D to C with capacity 1? What if there are no ClassA employees in d1?
- **- 1 pts** Direct all edges from D to C with capacity 1? What if there are no ClassA employees in d1? The Supply can not be considered as a source node.
- **- 1 pts** Pi means department? Then What if there are no ClassA employees in Pi?
- **- 0.3 pts** How can a staff connect to multiple classes
- **- 1 pts** Department can not belongs to X-class
- **- 1 pts** Need to combine the class node, and set appropriate weight
- **- 0.1 pts** Since no separate staff node is set, the result needs to be selected from the staff of the same class in the same department
- **- 1 pts** The capacity from s to Si should be 1.
- **- 1 pts** What if there are no ClassA employees in di?
- **- 1 pts** One staff can not belong to multiple departments.
- **- 0.5 pts** What if there are no ClassA employees in di?
- **- 0.5 pts** There must be m1 number of A-class staff members, m2 number of B-class staff members, m3 number of C-class staff members, and m4 number of D-class staff members
- **- 0.2 pts** staff should only link to corresponding department and class
- **- 0.2 pts** t node is undefined
- **- 0.5 pts** s and t node are undefined
- **- 0.5 pts** capacities are undefined
- **- 0.1 pts** Details are unclear
- **- 1 pts** Click here to replace this description.
- **- 0.2 pts** The capacity of source to class node is undefined
- **- 0.5 pts** Details are unclear
- **- 1 pts** Try maximum flow

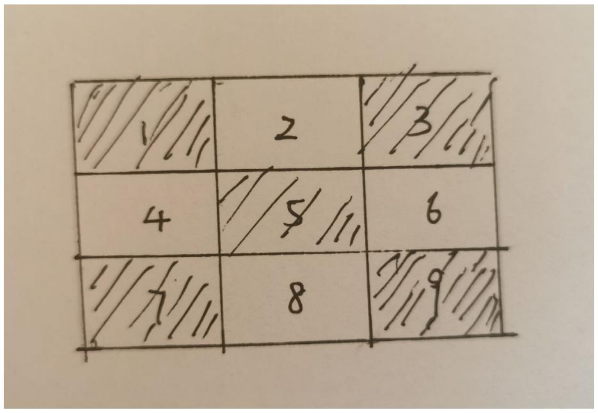


Then, run Ford-Fulkerson algorithm on this graph, then, if $departmentOf(s_i) \to s_i \to classOf(s_i)$ has $flow = 1$ in the max-flow, then s_i is selected. Hence, we have determined who should be selected, in $O(VE^2)$ time (BFS based), where $V = k + n + 6$, for k departments, n staff, 4 classes, s and t, and $E = k + 2n + 4$ for n edges representing $s \to d_i$, n edges representing $d_i \to s_j$, n edges representing $s_j \to class$, and 4 edges representing $class \to t$.

Problem 3

Solution

First, we divide the matrix into 2 parts, the first part contains $(2i-1, 2j-1)$ and $(2i, 2j)$, the other part contains $(2i-1, 2j)$ and $(2i, 2j-1)$, like the white grids and black grids in chessboard (shown in the following figure).

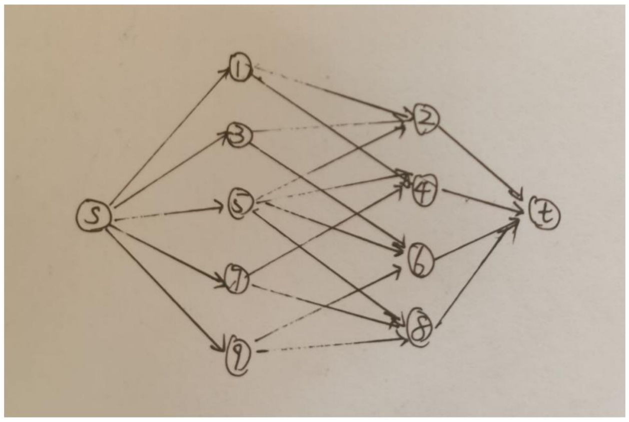


Then, we can construct a graph by following steps:

1. Add a source s connects to all part-I (black) points (1,3,5,7,9 in this figure), with capacity equals to the corresponding value in matrix($cap(s \to P_{ij}) = Mat[i, j]$).
2. Each part-I points connects to all neighboring part-II (white) points (i.e. 1 connects to 2 and 4, 5 connects to 2,4,6,8), with $capacity = INF$.

3. Each part-II points connects to t, with capacity equals to the corresponding value in matrix($cap(P_{ij} \rightarrow t) = Mat[i,j]$), as shown in the following figure.

Here is the graph of the example shown above:



Then, run Ford-Fulkerson algorithm and find the max-flow. By maxflow mincut theorem, the capacity of max-flow is also the capacity of the min-cut. That means, the edges which are chosen in the min-cut has the minimum sum. So, the remaining value is maximum sum.

$$MaxSum = \sum_i \sum_j M[i,j] - Mincut$$

Problem 4

Solution
First, run Ford-Fulkerson algorithm to get max-flow and residual graph G_f in polynomial time. Then let A be the set of vertice can be reached by s(source) in the G_f. Then, we can obtain that the cut (A, Ac) has the capacity of the max-flow, and is the min-cut (by maxflow-mincut theorem).
Consider the edges from nodes in A to nodes in A_c, if (A, A_c) is the unique min-cut, then we can increase the max-flow by increasing any one of these edge's capacity by 1; In there exists other min-cut, the capacity of min-cut will not change and so as the max-flow. Therefore, we can first give a naive algorithm to check its unique:

1. Iterately choose an edge from A to A_c;
2. Increase the capacity of the edge by 1, check if there is an augmenting path in the G'_f (notice each time we only change one edge's capacity, the capacity of former edges we chosen should restore to the origin);
3. If for everytime we cannot find augmenting path, (A, Ac) is the unique min-cut; otherwise once we found an augmenting path, (A, Ac) is not the unique min-cut.

This checking algorithm will choose E edges at most, and for each iteration, it will run a DFS to check if the min-cut changes, which will take $O(V + E)$ time. Thus, this algorithm has time complexity of $O(E(V + E))$, which is a polynomial time.

3 **1 / 1**

✓ **- 0 pts** Correct

- 0.4 pts Wrong graph

- 0.4 pts Wrong algorithm descriptions

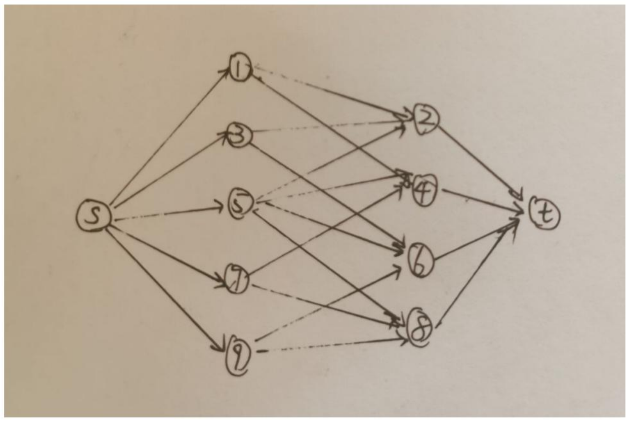
- 0.1 pts Wrong result

- 1 pts totally wrong

- 0.2 pts Inaccuracy of description

3. Each part-II points connects to t, with capacity equals to the corresponding value in matrix($cap(P_{ij} \rightarrow t) = Mat[i,j]$), as shown in the following figure.

Here is the graph of the example shown above:



Then, run Ford-Fulkerson algorithm and find the max-flow. By maxflow mincut theorem, the capacity of max-flow is also the capacity of the min-cut. That means, the edges which are chosen in the min-cut has the minimum sum. So, the remaining value is maximum sum.

$$MaxSum = \sum_i \sum_j M[i,j] - Mincut$$

Problem 4

Solution

First, run Ford-Fulkerson algorithm to get max-flow and residual graph G_f in polynomial time. Then let A be the set of vertice can be reached by s(source) in the G_f. Then, we can obtain that the cut (A, Ac) has the capacity of the max-flow, and is the min-cut (by maxflow-mincut theorem).

Consider the edges from nodes in A to nodes in A_c, if (A, A_c) is the unique min-cut, then we can increase the max-flow by increasing any one of these edge's capacity by 1; In there exists other min-cut, the capacity of min-cut will not change and so as the max-flow. Therefore, we can first give a naive algorithm to check its unique:

1. Iterately choose an edge from A to A_c;
2. Increase the capacity of the edge by 1, check if there is an augmenting path in the G'_f (notice each time we only change one edge's capacity, the capacity of former edges we chosen should restore to the origin);
3. If for everytime we cannot find augmenting path, (A, Ac) is the unique min-cut; otherwise once we found an augmenting path, (A, Ac) is not the unique min-cut.

This checking algorithm will choose E edges at most, and for each iteration, it will run a DFS to check if the min-cut changes, which will take $O(V + E)$ time. Thus, this algorithm has time complexity of $O(E(V + E))$, which is a polynomial time.

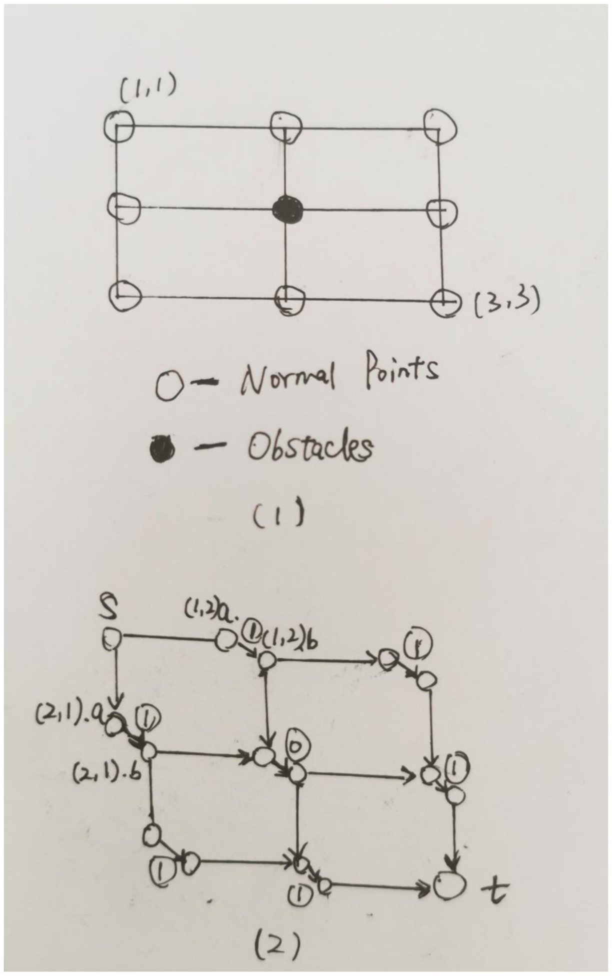
4 1 / 1

- ✓ **- 0 pts** Correct
- **- 0.5 pts** After iterating, no volume changes? Conflict with point 2
- **- 1 pts** Click here to replace this description.
- **- 0.1 pts** True and False are reversed
- **- 0.3 pts** [illegible]
- **- 0.3 pts** Lack of detail

Problem 5

Solution

We can construct a graph: Let $(1,1)$ be the source s and (m,n) be t. For each point (i,j), excepts $(0,0)$ and (m,n), we split it into two points $(i,j).a$ and $(i,j).b$, where $(i,j).a$ only connects to $(i,j).b$, and $(i,j).b$ connects to $(i,j+1).a$ and $(i+1,j).a$. Then, if (i,j) is not an obstacle, set the capacity of edge $(i,j).a \to (i,j).b$ to 1, otherwise set $(i,j).a \to (i,j).b$ to 0. The capacity of other edges are all set to 1 (actually any positive integer is OK). Then run Ford-Fulkerson algorithm on this graph, and find the maximum flow. The capicity of the max-flow is the number of obstacles needed. For example, in following case (1), the graph that we construct is shown in (2):



5 **1 / 1**

- ✓ **- 0 pts** Correct
- **- 0.8 pts** Wrong
- **- 0.5 pts** Wrong detail
- **- 1 pts** No answer
- **- 0.2 pts** Unclear description

Problem 6

Solution

1. We can solve this problem by using dynamic programming algorithm. Let $DP[i]$ be the length of longest ascending subsequence in $x_1, \dots, x_i$. Obviously the solution L we want to find is $DP[n]$. Consider the base case, there is only one item x_1, then, we can get $DP[1] = 1$ easily. Then we can obtain this recursive function:

$$DP[i+1] = \max\{DP[j]+1, For\ all\ 1 \le j \le i\ and\ x_j < x_{i+1}\}$$

   Hence, by the base case and recursive function, we can get $DP[n]$ in $O(n^2)$ time.

2. First, we should use dynamic programming algorithm described in (1) to find an array DP. Then, we can construct a graph by following steps:

   (a) Add source s and terminal t, use p_i represent i^{th} point.

   (b) For all $1 \le i \le n$, split p_i into 2 points $p_i.a$ and $p_i.b$, link $p_i.a$ and $p_i.b$ with an edge with $capacity = 1$.

   (c) For all $1 \le i \le n$, if $DP[i] == 1$, link s with $p_i.a$, with $capacity(s \to p_i.a) = 1$. If $DP[i] == L$, link p_{i+L} to t, with $capacity(p_{i+L}.b \to t) = 1$.

   (d) For all i, j satisfies $j < i$ and $x_j < x_i$ and $DP[j] + 1 = DP[i]$, then connect $p_j.b$ to $p_j.a$, with an edge of $capacity(p_j.b \to p_i.a) = 1$.

   Run Ford-Fulkerson algorithm on this graph, to find the max-flow. And the capacity of the max flow is the number of non-overlapping longest ascending subsequence.

6 **1 / 1**

- ✓ **- 0 pts** Correct
- **- 0.2 pts** Incorrect DP
- **- 0.2 pts** didn't split node, w(li,ri)=1
- **- 0.2 pts** Incorrect node connection.

w(ri,lj)=1 if j<i, x_j<x_i, dp(j)=dp(i)+1

- **- 0.2 pts** Incorrect source to node.

Only connect the node whose value in DP is one.

- **- 0.2 pts** Incorrect sink to node.

Only connect the node whose value in DP is L.

- **- 0.8 pts** Incorrect count subsequence