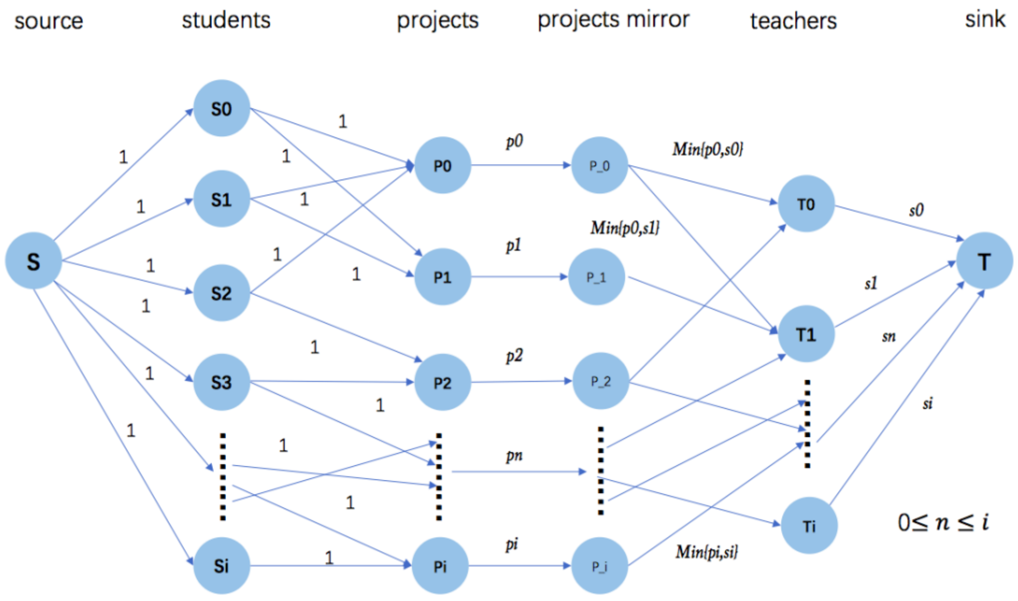
Task1

1. Transform the input

   Transform the input to a flow network G with an integer d, which represents the minimum cut in G. General Graph



Explanation: --all notations are quoted from assignment sheet

- Assign edges with unit capacity to each student vertex from source.
- Connecting each student with their preference projects according to w_i (preference projects for each students), and each edge has unit capacity.
- Projects mirrors vertices and edges between projects and projects mirror are used to restrict the flow can be pass through projects vertices, the capacities of those edges are based on p_i. (the number of student each project can be studied by)
- Connecting each teacher vertex with their preference projects (projects mirrors) according to t_i (preference projects for each teacher). Edges capacities are decided by Min{p_i,s_i}. If p_i is greater than s_i, because teacher can only supervise s_i student, the capacity has to be set s_i to avoid overflow. If pi is smaller than s_i, projects vertices can only pass through up to p_i flow, so the capacity has to be set p_i. If pi equals to s_i, we set the capacities p_i.
- All edges come from teachers vertices go to the sink and the capacities of each edge is based on s_i (the number of student each teacher can supervised)
- Integer d is the total number of students, because the capacities are all

based on p_i, s_i.

2. Solve the transform problem

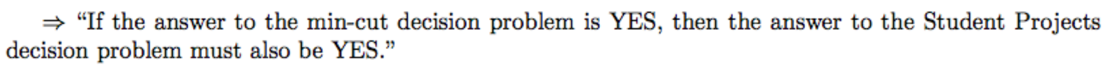
   Firstly, we should find the min-cut of the generating graph. To solving the min-cut and max-flow problem, Ford-Fulkerson is the best choice. In order to solve the problem efficiently, we use Edmonds-Karp to improve the original Ford-Fulkerson whose time complexity is O(mC), which means that if the upper bound of maximum flow is really large, the algorithm will cost huge time. Instead of using upper bound of max flow, Edmonds-Karp based on the number of edges and vertices $O(nm^2)$ is more effective. Secondly, we compare the capacity of min-cut with the integer d(the number of student) to find out the answer of the min-cut decision problem.

3. Transform the output

   The question (1) is asking about can we allocate projects such that **every student** has a project to study on. In question (2) source provides flow to pass through every vertex in graph G and reach the sink. If after running the algorithm, Max flow equals to the number of students from source, which means that every student has a path from sources to sink. According to the Max flow Min cut theorem mentioned in lecture, Max flow equals Min cut. Overall, if the d which is the lower boundary of min-cut can equal to the number of students, it means that the generating graph must allow every student to pass through it, which reflects that every student has a project to study on.

Task2

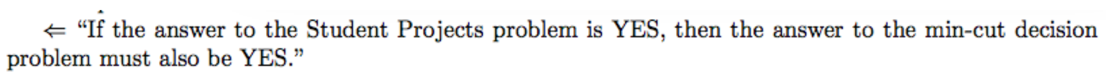
Theorem 1:



⇒ "If the answer to the min-cut decision problem is YES, then the answer to the Student Projects decision problem must also be YES."

If the answer to the min-cut decision problem is Yes, according to max-flow and min-cut theorem, max flow should be greater or equal to integer d. In specific, based on the student project decision problem to generating the corresponding graph with source and sink, we can decide that integer d is the number of students. Therefore, max flow of the graph at least equals to the number of students, implying that every student has a project to study on.

Theorem 2:



$\Leftarrow$ "If the answer to the Student Projects problem is YES, then the answer to the min-cut decision problem must also be YES."

If the answer to the Student Projects problem is YES, generating the corresponding graph, we can make sure that every student can pass from source to sink, which means that the max flow at least is the number of students. According to max-flow and min-cut theorem, max flow equals to min cut, we can decide that there exists an integer d which is the number of students is the lower boundary of the capacity of min-cut, in order words, the capacity of the min-cut of given graph G is at least d.

Because it is possible to allocate projects such that every student has a project to study on if and only if the capacity of the min-cut of given graph G is at least a given integer d, the reduction works.

Task3: -- all notations are quoted from assignment sheet

- Transform input

Referring the graph above, and notation of variables listed on specification

a) Creating source and sink costs 2*O(1)=O(1)

b) Creating n student vertices costs n*O(1)=O(n)

c) Creating L project vertices costs L*O(1)=O(L)

d) Creating L project mirror vertices costs L*O(1)=O(L)

e) Creating m teacher vertices costs m*O(1)=O(m)

f) Setting edges and each weight between source and student vertices costs n*(O(1)+O(1))=O(n)

g) Setting edges and each weight between students and projects costs n*sizeof(w_i)*(O(1)+O(1))=O(n*sizeof(w_i))

h) Setting edges and each weight between projects and projects mirror costs L*(O(1)+O(1))=O(L)

i) Setting edges and each weight between projects mirror and teachers costs m*sizeof(t_i)*(O(1)+O(1))=O(m*sizeof(t_i))

j) Setting edges and each weight between teacher vertices and sink costs m*O(1)=O(m)

n is the number of students, sizeof(w_i) is the size of preference projects of each student, m is the number of teachers, and sizeof(t_i) is the size of preference projects of each teacher.

- Time to take to solve the min cut and max flow problem

Using Edmonds-Karp Algorithm to calculate max flow and mini cut, which costs $O(NM^2)$. N is the number of vertices, and M is the number of edges. In this case, vertices include source, sink, student vertices, teacher vertices, project vertices, and project mirror vertices, so $\underline{N=1+1+n+m+2L}$. And the number of edges is $\underline{M=n+n*\text{sizeof}(w_i)+L+m*\text{sizeof}(t_i)+m}$

- Time to transform the output O(1)

Overall: $O(n)+O(L)+O(m)+O(n*\text{sizeof}(w_i))+O(m*\text{sizeof}(t_i)) + O((n+m+2L)*(n+n*\text{sizeof}(w_i)+L+m*\text{sizeof}(t_i)+m)^2)$