CS180 Homework 8

Zhehao Wang 404380075 (Dis 1B)

May 24, 2016

1 Min cut of the graph

Algorithm description: given the graph $G = (V, E)$, $V = v_1 ... v_n$; We first do $\frac{n}{2}$ MaxFlow between all the pairs $(v_1, v_2), (v_3, v_4)...(v_{n-1}, v_n)$, then $\frac{n}{4}$ MaxFlow between pairs $(v_1, v_3), (v_5, v_7)...$, ..., until the last step where we do 1 MaxFlow between the pair $(v_1, v_{\frac{n}{2}})$.

(At step i, we do MaxFlow between pairs $(v_1, v_{1+2^{i-1}}), (v_{1+2\cdot 2^{i-1}}, v_{1+3\cdot 2^{i-1}}), (v_{1+4\cdot 2^{i-1}}, v_{1+5\cdot 2^{i-1}})...$, and the number of MaxFlow is $\frac{n}{2^i}$)

We do $\Sigma_{i=1}^{\log_2 n} \frac{n}{2^i} = n$ total MaxFlow, and the max value among all of these MaxFlow is the min cut of the entire given graph.

Time complexity: with Ford Fulkerson algorithm[1], the complexity of each MaxFlow is $O($ the number of edges $\cdot$ the value of the MaxFlow). As we are given an unweighted graph, the max value of the MaxFlow is the number of edges m. Thus each MaxFlow is $O(m^2)$, and we do n in total, so the overall complexity is $O(m^2 n)$

Correctness: we want to show that the n pairs described above would cover the min cut of the graph (S, T) where S and T are two set of nodes divided by the min cut, since if so, the algorithm's correct by **MaxFlow-min-cut theorem**.

Both S and T are non-empty, and the min cut (S, T) is covered if at least one of our pairs (s', t') satisfies $s' \in S$ and $t' \in T$. Assume that none of our pairs satisfies the condition. Given the description above, we know at step 1 nodes (v_1, v_2) and nodes (v_3, v_4) are each in the same S or T, and in step 2, we know that nodes (v_1, v_3) are in the same S or T, thus combining the first two steps we have $v_1 ... v_4$ are in the same S or T. Similarly for each step, to satisfy our assumption, doing MaxFlow for $(v_1, v_{\frac{k}{2}})$ shows that nodes $v_1 ... v_k$ are in the same S or T. Until the last step we have $v_1 ... v_n$ are all in the same S or T, which contradicts with both S and T being non-empty.

Thus the min cut of the graph (S, T) will be covered by at least one pair of nodes among our n MaxFlow, and the algorithm's correct.

2 Menger's theorem for vertices

The theorem holds.

Proof: for the theorem we can consider only directed graphs, as undirected graph can be represented as directed graph with edges both way.

In the directed graph $G = (V, E)$, for each node $v \in V - \{s, t\}$, we split v into two nodes i, o connected by an edge (i, o), and all incoming edges to v go into i, all outgoing edges from v exit from o. Each edge has capacity 1. Do MaxFlow for the transformed graph G', in the MaxFlow each $v = (i, o)$ node

[1] Whose complexity we analyzed in class

would be used at most once, since each (i, o) edge has capacity 1. With this we transform the problem into **Menger's theorem** for edges [2], which is immediate from **MaxFlow-min-cut theorem**.

3 Deal cards and select

Solution: let undirected unweighted bipartite graph $G = (LeftV, RightV, E)$. For any dealing of cards, let each node in *LeftV* represent a pile of cards, and each node in *RightV* represent a rank. Add an edge between a node $s \in LeftV$ and a node $t \in RightV$ for each card of rank t in pile s.

In the bipartite graph G, each node $s \in LeftV$ has degree 4, and each node $t \in RightV$ has degree 4. So for each subset of nodes $S \subseteq LeftV$, $|N(S)| \geq |S|$ (Since otherwise there exists an S', such that the degree of $S' = 4 \cdot |S'| > 4 \cdot |N(S')|$, which contradicts with the definition of $N(S')$ since the total degree on S''s side is larger than that on $N(S')$'s side). Thus by **Frobenius Hall theorem**[3], there exists a perfect matching between *LeftV* and *RightV*. And selecting a card of rank t in pile s only if the edge (s, t) is in the perfect matching shows the conclusion.

4 Exam scheduling max flow

Solution: let undirected unweighted bipartite graph $G = (LeftV, RightV, E)$. For any classes, rooms and times combination, let each node $s_i \in LeftV$ represent a class E_i, and each node $t_{jk} \in RightV$ represent a room S_j at time T_k. Add an edge between a node $s_i \in LeftV$ and a node $t_{jk} \in RightV$ only if $E_i < S_j$ (class size smaller than room size).

The maximum number of exams that can be scheduled is the max matching (of edges that share no vertices) in bipartite graph G. To solve the max matching in bipartite graph[4], we add a node a that connects to all $s_i \in LeftV$, and a node b that connects to all $t_{jk} \in RightV$, set the capacity of the newly added edges to 1, the capacity of the already existing edges to ∞, and do MaxFlow from a to b.

A schedule exists iff the MaxFlow equals with the number of classes; For each edge (s_i, t_{jk}) in the max matching, we assign the class E_i to room S_j at time T_k.

Time complexity: the algorithm is $O(n^2rt)$, since the max possible value of MaxFlow is n, and max number of edges in the graph is $n \cdot r \cdot t$. Similar with the analysis of the complexity of Ford Fulkerson algorithm, the complexity of this solution is $O(n^2rt)$.

[2] Which we proved in class
[3] Which we proved in class
[4] Which we talked about in class