CSC373 Fall 2017 Assignment 2

1. Suppose we have a maximum integral flow f with $val(f) \geq 1$ in a flow network $\mathcal{F} = (G, s, t, c)$ with integral capacities.
   a) Does there always exist an edge e such that by decreasing the capacity of e by one unit to $c(e) - 1$, the value of the maximum flow is decreased by exactly one unit? Justify your answer.
      - **From the question we can see that, when if we decreasing the capacity of an edge means that we decrease the capacity of any cut by at most one.**
      - **Then, the value of the maximum flow is decreased by one, therefore at most one.**
      - **Let say that there is an edge e = (u, v), which its capacity was decreased by one,**
      - **And there is a previous flow f', such that it was decreased by one to the current one.**
      - **Then compute a shortest path between s and u such that it has a positive flow and it will decrease the flow by 1.**
      - **Then compute another path between u and t with such that it has a positive flow and it also decrease the flow by 1.**
      - **Then by now, in the current iteration f, the flow should be one less than the value in iteration f'.**
      - **Therefore, if we are finding an augmented path in this graph, if there is no path like this, that means f has the maximum flow**
      - **if there exists a path like that, augmented it by one and then it will decrease at most one and return to the maximum flow.**

   b) Does your answer to the above question depend on the assumption that all capacities are integral?
      - **Yes, by the Integral Theorem, if the capacities of the flows are integral, then there exists an integral maximal flow.**
      - **Since every augmenting path increases flows by an integer amount, then the capacities must be integral too.**

   c) Suppose that there is an edge e such that increasing its capacity by one unit will result in the increase of the maximum flow value. Show how to find such an edge in time that is determined by the computation of a least cost paths algorithm.
      - **Same train of thoughts as of part a), we can see that the value of the new max flow is either the one from the previous iteration, or it will increase by at most one.**
      - **That is because that the value of any minimum cut can only increase by one, so the value of the max flow can only increase by at most one.**
      - **So, if we increase the capacity of an edge e by one, and we are trying to find an augmenting path in the the new graph:**
      - **if there is no augmenting path like this, that means e already has the maximum flow**
      - **if there is an augmenting path, augment along it.**
      - **Since the capacities are integral, the flow will be augmented by one, and it will be the max flow.**

2. Consider the following makespan problem in the restricted machines model. The input is a set $\mathcal{J}$ of unit processing time jobs $\{J_1, \ldots, J_n\}$ where each job J_j can be scheduled on some subset $S_n \subseteq \mathcal{M} = \{M_1, \ldots, M_m\}$ of the m identical machines. A schedule is a mapping $\sigma : \mathcal{J} \to \mathcal{M}$ such that $\sigma(J_j) = M_i$ implies $M_i \in S_j$. The objective is to compute a schedule σ so as to minimize the makespan value which is $max_i|j : \sigma(J_j) = M_i|$. That is, to minimize the latest completion time (= number of jobs) for all machines.
   a) Show that the online greedy algorithm is not optimal.
      - **If there are jobs {J_1, J_2, J_3, J_4, J_5, J_6} and machines {M_1, M_2, M_3}**

- **If we are using the online greedy algorithm,**
- **Then $\sigma(J_1) = M_1$, $\sigma(J_2) = M_2$, $\sigma(J_3) = M_3$, $\sigma(J_4) = M_1$, $\sigma(J_5) = M_1$, $\sigma(J_6) = M_1$**
- **Then M_1 will finish at last and 3 jobs.**
- **But the optimal solution should be $\sigma(J_1) = M_1$, $\sigma(J_2) = M_2$, $\sigma(J_3) = M_3$, $\sigma(J_4) = M_1$, $\sigma(J_5) = M_2$, $\sigma(J_6) = M_2$**
- **Then all M_1, M_2 and M_3 will finish at the same time with 2 jobs each.**
- **Therefore, the online greedy algorithm is not optimal.**

b) Show how to optimally solve this makespan problem by reducing the problem to optimal flows
- **Let us first construct a flow network F = (G, s, t, c)**
- **And G is a graph which it contains vertices and edges such that G= (V, E),**
- **which E = {(s, m), m∈M}∪{(j, t), j∈J}∪{(m, j), σ(J) = m, j∈J, m∈M}**
- **If (m, j)∈E, j∈J, m∈M, then that means c(m, j) = 1**
- **If (j, t)∈E, j∈J, then that means c(j, t) = 1**
- **If (s, m)∈E, m∈M, then that means c(s, m) = $\frac{\lceil number\ of\ machines \rceil}{number\ of\ jobs}$**
- ***Now we can get the max flow f in F using Ford-Fulkerson algorithm**
- **If val(f) = number of jobs, then we can get the optimal schedule σ where if f(m, j) = 1, j∈J, m∈M, optimal makespan value = max f(s,m). ***
- **Else, c(s, m) = c(s, m) + 1, for all (s, m)∈E, m∈M**
- **Then we keep repeating the steps between the two *'s.**

- **Since all the capacities are integral, then the steps between the two *'s will always terminate with an optimal integral max flow.**
- **The steps between the two *'s will eventually terminate when c(s, m) becomes large enough so that f max flow = min cut in F = sum of c(j, s), j∈J = number of jobs.**
- **Once terminate, c(s, m) is the makespan value closest to $\frac{\lceil number\ of\ machines \rceil}{number\ of\ jobs}$, then c(s, m) is the optimal makespan value.**

3. Consider the graph 3-colourability problem.
   a) Show how to reduce the search problem to the decision problem. That is, how to find a valid 3-colouring (when one exists) given that there is a subroutine for determining if a graph has a 3-colouring.
   - **Basecase: we need to check if the graph G can be 3-coloured, if not, we do not need to proceed further in the question. We have reached the end; therefore, we can stop right here.**
   - **Else, we need to find the colour of the graph G.**
   - **By the hint: we will create the colouring of one node at a time**
   - **So we need to colour one node at a time, so that at any time, at least a part of the solution can be a 3-colouring of all the other nodes in the graph G.**
   - **Let's call the three colours C_1, C_2, C_3.**
   - **And add the edges between all three of them**
   - **Then, now they are forming a triangle and add the triangle to the graph G.**
   - **Let's call the new graph G'**
   - **To create the colouring of one node of the graph, say node u, we need to add edges to G' between u and other 2 nodes in C_1, C_2, C_3.**
   - **If we have added the edges (u, C_1) and (u, C_2), then we are colouring the node u with C_3.**
   - **We assume that there are a sequence of graphs G_1, …., G_n where they are all 3-colourable and G_i is obtained from its previous graph G_{i-1}**
   - **And $G_i = G_{i-1}$ + (u_i, C_1)+ (u_i, C_2), if this G_i is 3 colourable,**
   - **Then if there are edges between (u_i, C_1) and (u_i, C_2), then u_i must be C_3**

- **if there are edges between (ui, C_1) and (ui, C_3), then u_i must be C_2**
- **if there are edges between (ui, C_2) and (ui, C_3), then u_i must be C_1**

b) Assume that your subroutine for the decision problem takes time $T(n, m)$ where $n = |V|, m = |E|$. What is the asymptotic time required for the search algorithm?
- **I do not the answer to this question.**

4. Consider the following Set Selection decision problem:
Input: Given an integer $k \geq 1$ and a collection of sets $\{C_1, \dots, C_n\}$ where each $C_i \subseteq \{1, 2, \dots, m\}$.
Decision problem: Does there exist a subset of indices $S \subseteq \{1, 2, \dots, n\}$ such that $|S| = k$ and $C_i \cap C_j = \emptyset$ for all $i, j \in S, i \neq j$?
Show that Set Selection is NP complete by showing that it is in NP and that it is NP hard by giving a polynomial time transformation from Independent Set.
- **Let us put this problem into a graph where each set is a vertex**
- **There is only an, edge between two vertices(sets) if they have an empty intersection.**
- **Organize the graph into connected components,**
- **let the sizes of the connected components be S_1,…,S_i.**
- **There would only exists a solution if there is a subset of S_1,…,S_i that has the sums of k.**
- **Then, this question can be solved in polynomial time.**
- **Therefore, it is NP-hard.**

5. Consider the following Unit Value-One Machine scheduling decision problem:
Input: Given an integer $k \geq 1$ and a collection of *unit value* jobs $\{J_1, \dots, J_m\}$ where each job J_i is described by a triple of integers(r_i, p_i, d_i) where $r_i \geq 0$ is the release time, p_i is the processing time, and d_i is the deadline for the job. That is, if J_i is scheduled then it must be scheduled to start at some time $t_i : r_i \leq t_i \leq d_i - p_i$.
Decision problem: Does there exist a subset S of k jobs that can be scheduled within their deadlines on one machine so that no two jobs in S are overlapping when scheduled?
a) Assuming $r_i = 0$ for all i, show that this problem can be solved by a greedy algorithm.
- **We should first sort all the hobs by their deadlines, from the earliest to the latest**
- **Let S be the set of jobs scheduled already and initialize S as S = {} and S' represents the set of jobs from the previous iteration.**
- **For each job j,**
- **Put j in S such that S = S' + j**
- **If j does not meet its deadline, then get rid of the job in S with the largest processing time.**
- **The rest of jobs can be kept in S, because we know that all the jobs in S before inserting j can meet their deadlines for sure.**
- **By the end, we can get a S as a set with the maximum jobs can be scheduled to process before their deadlines.**
- **Let us compare the number of jobs in S with k**
- **If it is greater or equal to k, then S exists,**
- **Otherwise, there is no such subset.**

- **We can prove by induction:**
- **Let S_n be the set of jobs scheduled by the end of iteration n**
- **Show that S_n is the schedule with the max sized subset of $\{J_1, \dots, J_n\}$, where $0 \leq n \leq m$**
- **Basecase: n = 0, then $S_n = 0$, therefore we scheduled 0 jobs**
- **Assume that in iteration n, S_n has scheduled the max sized subset of $\{J_1, \dots, J_n\}$**
- **For the iteration n+1, we are trying to fit J_{n+1} into S_{n+1}**
- **If J_{n+1} can be finished before its deadline, $S_{n+1} = S_n$**

- **If J_{n+1} cannot be finished before its deadline, we delete the job in the current S_n with the longest processing time and let $S_{n+1} = S_n$ – longest_processing_job**
- **J_{n+1} can be scheduled because it has the latest deadline among all the other jobs in S_n.**
- **Since S_n has the max number of jobs in $\{J_1, ..., J_n\}$**
- **And $S_n \cup \{J_{n+1}\}$ cannot be scheduled,**
- **The max number of jobs in $\{J_1, ... , J_n, J_{n+1}\}$ can be scheduled equals to the size of S_n.**
- **Then size of S_{n+1} = size of S_n**
- **Then S_{n+1} does have the max sized subsets of $\{J_1, ..., J_{n+1}\}$**
- **Therefore, by induction, S_n is the schedule with the max sized subset of $\{J_1, ..., J_n\}$, where $0 \leq n \leq m$.**

b) Show that when the release times are part of the problem (i.e. not assuming that all $r_i = 0$), that the Unit Value-One Machine scheduling problem is NP complete.
- **As given that we have a subset S with k number of jobs, and it is ordered by their deadlines.**
- **It is always easy for us to check whether if every job can meet its deadline**
- **Then it takes polynomial time to check is it is correct.**
- **Then, the Unit Value-One Machine problem is NP complete.**