CS 170 HW 12

Due **2021-11-22, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

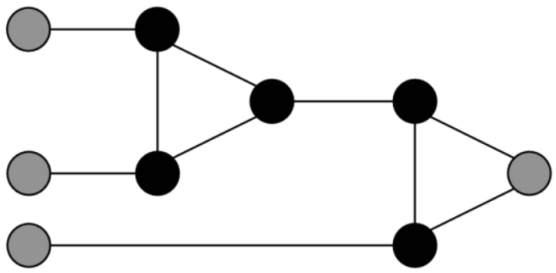
2 Reduction to 3-Coloring

Given a graph $G = (V, E)$, a valid 3-coloring assigns each vertex in the graph a color from $\{0, 1, 2\}$ such that for any edge (u, v), u and v have different colors. In the 3-coloring problem, our goal is to find a valid 3-coloring if one exists. In this problem, we will give a reduction from 3-SAT to the 3-coloring problem. Since we know that 3-SAT is NP-Hard (there is a reduction to 3-SAT from every NP problem), this will show that 3-coloring is NP-Hard (there is a reduction to 3-coloring from every NP problem).

In our reduction, the graph will start with three special vertices, labelled "True", "False", and "Base", and the edges (True, False), (True, Base), and (False, Base).

(a) For each variable x_i in a 3-SAT formula, we will create a pair of vertices labeled x_i and $\neg x_i$. How should we add edges to the graph such that in any valid 3-coloring, one of $x_i, \neg x_i$ is assigned the same color as True and the other is assigned the same color as False?

(b) Consider the following graph, which we will call a "gadget":



Show that in any valid 3-coloring of this graph which does not assign the color 2 to any of the gray vertices, the gray vertex on the right is assigned the color 1 only if one of the gray vertices on the left is assigned the color 1.

(c) We observe the following about the graph we are creating in the reduction:

(i) For any vertex, if we have the edges (v, False) and (v, Base) in the graph, then in any valid 3-coloring v will be assigned the same color as True.

(ii) Through brute force one can also show that in the gadget, for any assignment of colors to gray vertices such that:

(1) All gray vertices are assigned the color 0 or 1

(2) The gray vertex on the right is assigned the color 1

(3) At least one gray vertex on the left is assigned the color 1

Then there is a valid coloring for the black vertices in the gadget.

Using these observations and your answers to the previous parts, give a reduction from 3-SAT to 3-coloring. Prove that your reduction is correct.

3 Multiway Cut

In the multiway cut problem, we are given a graph $G(V, E)$ with k special vertices $s_1, s_2 \dots s_k$. Our goal is to find the smallest set of edges F which, when removed from the graph, disconnect the graph into at least k components, where each s_i is in a different component. When $k = 2$, this is exactly the min s-t cut problem, but if $k \geq 3$ the problem becomes NP-hard.

Consider the following algorithm: Let F_i be the set of edges in the minimum cut with s_i on one side and all other special vertices on the other side. Output F, the union of all F_i. Note that this is a multiway cut because removing F_i from G isolates s_i in its own component.

(a) Explain how each F_i can be found in polynomial time.

(b) Let F^* be the smallest multiway cut. Consider the components that removing F^* disconnects G into, and let C_i be the set of vertices in the component with s_i. Let F_i^* be the set of edges in F^* with exactly one endpoint in C_i. How many different F_i^* does each edge in F^* appear in? How do the sizes of F_i and F_i^* compare?

(c) Using your answer to the previous part, show that $|F| \leq 2|F^*|$. (Challenge: How could you modify this algorithm to output F such that $|F| \leq (2 - \frac{2}{k})|F^*|$?)

(As an aside, consider the minimum k-cut problem, where we want to find the smallest set of edges F whose removal disconnects the graph into at least k components. The following greedy algorithm for minimum k-cut gets a $(2 - \frac{2}{k})$-approximation: Initialize F to the empty set. While $G(V, E - F)$ has less than k components, find the minimum cut within each component of $G(V, E - F)$, and add the edges in the smallest of these cuts to F. Showing this is a $(2 - \frac{2}{k})$-approximation is fairly difficult.)

4 Project

Please answer the following questions after reading the project specification. They will help you build intuition for how to construct your inputs and solvers in the project.

(a) Construct a set of igloos polishing tasks such that the problem, as specified in the project specifications, has a trivial solution. Explain why the input is trivial.

(Hint 1: Your answer should focus on the relationship between the parameters of tasks, instead of the actual numerical values.)

(Hint 2: Try setting the deadlines of all tasks to 1440.)

(b) For this question **do not** use the profit decay function specified in the specification. Instead assume that the profit for polishing is all or nothing. If an igloo is polished by the deadline, the profit gained is equal to the corresponding p_i. Otherwise if an igloo is polished after the deadline, then the profit gained is 0.

Next, find the best sequence of igloos to polish in a total of **100 minutes**, given the following igloo polishing tasks:

- Igloo 1: Duration = 30, Deadline = 65, Profit = 10
- Igloo 2: Duration = 40, Deadline = 100, Profit = 20
- Igloo 3: Duration = 15, Deadline = 80, Profit = 5
- Igloo 4: Duration = 20, Deadline = 30, Profit = 30
- Igloo 5: Duration = 25, Deadline = 40, Profit = 15

(i) Find the optimal sequence of igloos to polish

(ii) If all igloos had identical deadlines, equal to 100, would that change the optimal sequence? Why?

(c) For this question **do not** use the profit decay function specified in the specification. Instead, assume that our profit decays according to the following function, $max(0, p_i - s_i)$, where p_i is the original profit and s_i is the number of minutes by which the task was delayed past the deadline.

(i) Consider a greedy solver, where the task with the **highest benefit** is chosen at every step and scheduled as soon as possible, until we run out of time or tasks to schedule. Construct an input, of exactly 2 igloo polishing tasks, such that the greedy solver chooses a sub-optimal list of igloos. Explain why the greedy algorithm fails.

(ii) Consider a greedy solver, where the task with the **next earliest deadline** is chosen at every step, until we run out of time or tasks to schedule. Construct an input, of exactly 2 igloo polishing tasks, such that the greedy solver chooses a sub-optimal list of igloos. Explain why the greedy algorithm fails.

5 (Extra Credit) Approximation Hardness of Independent Set

Recall the maximum independent set problem: Given an undirected graph G, we want to find the largest independent set, i.e. largest set of vertices S such that no two vertices in S are adjacent. An algorithm is a α-approximation algorithm for maximum independent set if

given a graph G, it outputs an independent set of size at least OPT/α, where OPT is the size of the largest independent set.

Show that if there is a polynomial-time $2^{2^{100}}$-approximation algorithm for maximum independent set, there is a polynomial-time 2-approximation algorithm for maximum independent set.