CS170–Spring 2019 — Homework 8 Solutions

Ran Liao, SID 3034504227

March 20, 2019

Collaborators:Jingyi Xu, Renee Pu

1 Study Group

Name	SID
Ran Liao	3034504227
Jingyi Xu	3032003885
Renee Pu	3032083302

2 Modeling: Tricks of the Trade

(a)

$$\min z_1 + z_2 + \cdots + z_n$$
$$z_i \geq y_i - (a + bx_i) \text{ for } i = 1, \cdots, n$$
$$z_i \geq -(y_i - (a + bx_i)) \text{ for } i = 1, \cdots, n$$

(b)

$$\min z$$
$$z \geq y_i - (a + bx_i) \text{ for } i = 1, \cdots, n$$
$$z \geq -(y_i - (a + bx_i)) \text{ for } i = 1, \cdots, n$$

3 Zero Sum Games

(a) x_1 is the probability that Alice will play strategy 1.

x_2 is the probability that Alice will play strategy 2.

p is Alice's payoff

(b)

$$\max p$$
$$p \leq 4x_1 + 2x_2$$
$$p \leq x_1 + 5x_2$$
$$x_1 + x_2 = 1$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$
$$p \geq 1$$

(c)

$$\max p$$
$$p \leq 4x_1 + 2(1 - x_1) = 2x_1 + 2$$
$$p \leq x_1 + 5(1 - x_1) = -4x_1 + 5$$
$$0 \leq x_1 \leq 1$$
$$p \geq 1$$

(d)

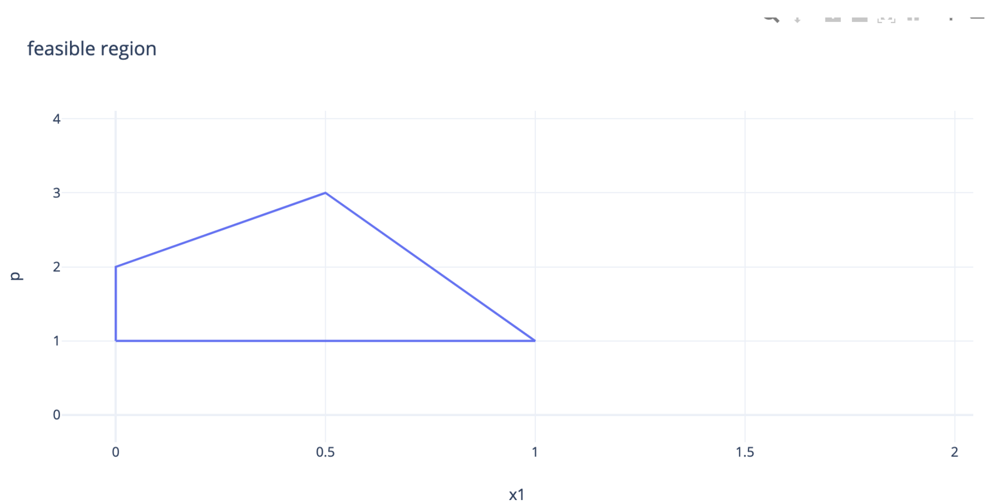

Figure 1: Feasible Region

(e) The optimal solution is $(\frac{1}{2}, \frac{1}{2})$ and the value of game is 3.

4 Repairing a Flow

(a) **Main Idea**

First sort all points into clockwise order and denote $D[i][j]$ as the distance between ith point and jth point, i.g. $D[i][j] = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Define subproblem as $T[i][j]$, which represents the optimal solution to triangulate polygon defined by ith point to jth point. The recursive relation is as follows. In base case, if $j - i < 3$, $T[i][j]$ is 0.

$$T[i][j] = \min\{\min_{k=i+2}^{j-2}\{T[i][k]+T[k][j]+D[i][k]+D[j][k]\}, T[i][j-1]+D[i][j-1], T[i+1][j], +D[i+1][j]\}$$

And the final solution is $T[1][n]$.

(b) **Proof of Correctness**

In base case, triangle do not need to be triangulated, so the cost is 0. In induction step, for $T[i][j]$, we have $j - i - 1$ ways to choose an vertex k, so that breaks it into two or one subproblems. One subproblem is triangulate polygon defined by ith point to kth point. The other is triangulate polygon defined by kth point to jth point. Choose the smallest cost among them can guarantee the optimality of $T[i][j]$. Therefore the final answer $T[1][n]$ is optimal.

(c) **Runtime Analysis**

There're n^2 subproblems and each of them will cost $O(n)$ time to check each possible situation. Therefore, the overall runtime is $O(n^3)$.

5 Three Partition

(a) **Main Idea**

Define subproblem as $X[i][s_1][s_2]$, which represents whether it's possible to divide first i numbers into three groups, such that the sum of first group is s_1 and the sum of second group is s_2. Suppose the ith number is $A[i]$. The recursive relation is as follows:

$$X[i][s_1][s_2] = X[i-1][s_1][s_2] \vee X[i-1][s_1 - A[i]][s_2] \vee X[i-1][s_1][s_2 - A[i]]$$

And the final solution is $X[n][\frac{total}{3}][\frac{total}{3}]$.

(b) **Proof of Correctness**

In base case, $X[1][0][0]$, $X[1][A[1]][0]$ and $X[1][0][A[1]]$ are true. This is trivial. In induction step, for $X[i][s_1][s_2]$ we have three choices, i.g. put ith number into first group, second group or third group. $X[i-1][s_1][s_2]$ is the situation that put it into first group. $X[i-1][s_1 - A[i]][s_2]$ is the situation that put it into second group. $X[i-1][s_1][s_2 - A[i]]$ is the situation that put it into third group. So $X[i][s_1][s_2]$ will be optimal and the final answer will be optimal.

(c) **Runtime Analysis**

There're $n(\sum a_i)^2$ subproblems and each of them will cost $O(1)$ time. Therefore, the overall runtime is $O(n(\sum a_i)^2)$.

6 2-SAT

(a) if G_I has a strongly connected component containing both x and $\neg x$ for some variable x. Then there's a path from x to $\neg x$ and a path from $\neg x$ to x. The edges in this graph can be considered as implication, so we have $x \Rightarrow \neg x$ and $\neg x \Rightarrow x$ by transitive rule. So if x is assigned with true, we have contradiction $true \Rightarrow false$. If x is assigned with false, we have $true \Rightarrow false$ as well. Therefore this problem has no valid solution.

(b) Note that the clause $(\alpha \vee \beta)$ is equivalent to $(\neg\alpha \Rightarrow \beta) \wedge (\neg\beta \Rightarrow \alpha)$. The edges added in to graph G_I is symmetric.

Suppose SCC A contains variables $v_1, v_2, v_3, \cdots v_n$. By symmetric property, there exists SCC $\neg A$ that contains variables $\neg v_1, \neg v_2, \neg v_3, \cdots \neg v_n$. Similarly, if there's an edge from SCC A to SCC B, there exist an edge from SCC $\neg B$ to $\neg A$.

In base case, there're only two SCCs and denote them as A and $\neg A$. This is trivial. Assigned one of them with true and the other with false will satisfy requirement. In induction step, suppose SCC A is a sink. Then SCC $\neg A$ must be a source. Suppose there's an edge from SCC B to SCC A. Then there's an edge from SCC $\neg A$ to SCC $\neg B$. Assign A with all trues can guarantee edge from SCC B to SCC A is satisfied. And assign $\neg A$ with all false can guarantee edge from SCC $\neg A$ to SCC $\neg B$ is satisfied. Therefore, delete all of them will not introduce conflicts and can always find a satisfaction solution.

(c) So first compute SCCs (meta-graph G_M of G_I) and assign value according to the previous mentioned way. Compute meta graph will cost $(O(|E| + |V|))$ time and assign will cost linear time. So the total runtime is linear.