Assignment 6

id: 519021910861

name: huidong xu

`date: 2021-12-13`

11-1

1. Let $w_1, w_2, w_3 = 1, 2, 1$, and $K = 2$. Then the greedy algorithm here will use three trunks, whereas there is a way to use just two.
2. Let $W = \sum_i w_i$. Note that in any solution, each truck holds at most K units of weight, so $\frac{W}{K}$ is a lower bound on the number of trucks needed. Suppose the number of trucks used by our greedy algorithm is an odd number $m = 2q + 1$. (The case when m is even is essentially the same, but a little easier.) Divide the trucks used into consecutive groups of two, for a total of $q + 1$ groups. In each group but the last, the total weight of containers must be strictly greater than K(else, the second truck in the group would not have been started then). Thus, $W > qK$, and so $\frac{W}{K} > q$. It follows by our argument above that the optimum solution uses at least $q + 1$ trucks, which is within a factor of 2 of $m = 2q + 1$.

11-3

1. Let $a_1, a_2 = 1, 100$, and consider the bound $B = 100$. Only a_1 will be chosen, while an optimal solution would choose a_2.
2. In fact, this can be done in $O(n)$ time. We first go through all the numbers a_i and delete any whose value exceeds B, such numbers cannot be used in any solution, including the optimal one, so we have not changed the value of the optimal by doing this. We then go through the numbers $a_1, a_2, \ldots, a_n$ in order until the sum of numbers we've seen so far first exceeds B. Let a_j be the number on which this happens. Thus we have $\sum_{i-1}^{j} a_i \geq B$, but

$\sum_{i-1}^{j-1} \leq B$and also $a_j \leq B$. Thus, one of the sets $a_1, a_2, \ldots, a_{j-1}$ or a_j is at least $\frac{B}{2}$ and at most B; we select this set as our solution. Since the optimum has sum of at most B, our solution is at least half the optimal value.

11-5

In the textbook we prove that

$$T - t_j \leq T^*$$

where T is our completion time, t_j is a size of a job and T^* is the optimal completion time. We also proved that the optimal completion time is at least the average load, which is at least 300 in our case

$$T^* \geq \frac{1}{m} \sum_j t_j \geq \frac{1}{10} \times 3000 = 300$$

We also know that $t_j \leq 50$. Therefore the ratio of difference between our completion time and the optimal completion time to the optimal completion time is at most

$$\frac{T - T^*}{T^*} \leq \frac{t_j}{T^*} \leq \frac{50}{300} = \frac{1}{6} \leq 20\%$$

13-1

As this is a maximization problem, we need an upper bound of c^*, and there is an easy one

$$c^* \leq m$$

where $m = |E|$.
The algorithm is: coloring every node independently with one of the three colors, each with probability $\frac{1}{3}$.
Let random variable

$$X_c = \begin{cases} 1 & edge\ e\ is\ satisfied \\ 0 & otherwise \end{cases}$$

Then for any giben edge e, there are 9 ways to color its two ends, each of which appears with the same probability, and 3 of them are not satisfying.

$$Exp[X_e] = Pr[e\ is\ satisfied] = \frac{6}{9} = \frac{2}{3}$$

Let Y be the random variable denoting the number of satisfied edges, then by linearity of expectations,

$$Exp[Y] = Exp[\sum_{e \in E} X_e] = \sum_{e \in E} Exp[X_e] = \frac{2}{3}m \geq \frac{2}{3}c^*$$

13-2

Number the voters $1, 2, \ldots, 100000$, where voters 1 through 20000 are the Republican voters. Let X_i be the random variable equal to 0 if i votes for R, and 1 if i votes for D. So $X = \sum_{i=1}^{100000} X_i$.

Now, for $i \leq 20000$, $EX_i = 0.99 \times 0 + 0.01 \times 1 = 0.01$. For $i > 20000$, $EX_i = 0.01 \times 0 + 0.99 \times 1 = 0.99$. By linearity of expectation,

$$EX = \sum_{i=1}^{100000} EX_i - 20000 \times 0.01 + 80000 \times 0.99 = 79400$$

13-3

1. Assume that using the described protocol, we get a set S that is not conflict free. Then there must be 2 processes P_i and P_j in the set S that both picked the value 1 and are going to want to share the same resource. But this contradicts the way our protocol was implemented, since we selected processes that picked the value 1 and whose set of conflicting processes all picked the value 0. Thus if P_i and P_j both picked the value 1, neither of them would be selected and so the resulting set S is conflict free. For each process P_I, the probability that it is selected depends on the fact that P_i picks the value 1 and all its conflicting processes pick the value 0. Thus $P[P_i\ selected] = \frac{1}{2} \times (\frac{1}{2})^d$. And since there are n

processes that pick values independently, the expected size of the set S is $n \times (\frac{1}{2})^{d+1}$.

2. Now a process P_i picks the value 1 with probability p and 0 with probability $1 - p$. So the probability that P_i is selected (i.e. P_i picks the value 1 and its d conflicting processes pick the value 0) is $p \times (1 - p)^d$. Now we want to maximize the probability that a process is selected. Using calculus, we take the derivative of $p(1 - p)^d$ and set it equal to 9 to solve for the value of p that gives the objective it's maximum value. The derivative of $p(1 - p)^d$ is $(1 - p)^d - dp(1 - p)^{d-1}$. Solving for p, we get $p - \frac{1}{d+1}$. Thus the probability that a process is selected is $\frac{d^d}{(d+1)^{d+1}}$ and the expected size of the set S is $n \times \frac{d^d}{(d+1)^{d+1}}$. Note that this is $\frac{n}{d} \times (1 - \frac{1}{d+1})^{d|1}$ and this later term is $\frac{1}{e}$ in the limit and so by changing the probability, we got a fraction of $\frac{n}{d}$ nodes. Note that with $p = 0.5$, we got an exponentially small subset in terms of d.