Homework #4

Due Time: 2019/01/03 14:20
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **three programming problems** and **three hand-written problems**, and the homework set including bonus are worthy of ≤ 120 points. If you get more than 100 points, your score will still be counted as 100 points.
- **Programming.** The judge system is located at `https://ada18-judge.csie.org`. Please login and submit your code for the programming problems (i.e., those containing "Programming" in the problem title) by the deadline. NO LATE SUBMISSION IS ALLOWED.
- **Hand-written.** For other problems (also known as the "hand-written problems"), you **MUST** turn in **a printed/written version** of your answers to the instructor at the beginning of the class on 2019/01/03 14:20. **Remember to print your name/student ID on the first page of your submitted answers.** You can also upload your homework to the NTU COOL system; however, it will be marked **only when** you have turned in the printed/written answer but it is lost during the grading. NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** Discussions with others are strongly encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (e.g., the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem. You may get zero point due to the lack of references.
- With TAs' discretion, too complicated solutions will be considered wrong.

Problem 1 - Knapsack 90% (Programming) (15 points)

Problem Description

Please approximately solve the *0/1 Knapsack* problem. Given n items with values $v_1, v_2, \ldots, v_n$ and weights $w_1, w_2, \ldots, w_n$ along with a knapsack capacity W, compute a subset of items with almost the maximum total value satisfying the capacity constraint.

Your answer will be considered correct if the total value in your knapsack is at least 0.9 times the total value in an optimal knapsack.

Input

The first line of input contains two space-separated integers n and W. Each of the following n lines has two space-separated integers v_i, w_i indicating the value and weight of the ith item.

- $1 \leq n \leq 1,000$
- $1 \leq W \leq 10^9$
- $1 \leq v_i, w_i \leq 10^9$

Output

Output k denoting the size of your knapsack in the first line. Output k space separated integers $a_1, a_2, \ldots, a_k$ in the second line denoting indices of items you chose.

Subtask 1 (50 %)

- $nW \leq 10^8$

Subtask 2 (50 %)

- no other constraints.

Sample Input

```
4 4
1 2
3 4
4 3
2 1
```

Sample Output

```
2
3 4
```

Hint

- Finish Problem 4. first.

Problem 2 - Yet Another Knapsack Problem (Programming) (20 points)

Problem Description

As an outstanding student, you already solved the previous problem easily. However, we know that you are not satisfied with an "approximated" solution. Thus, here is the *0/1 Knapsack* problem that you have to find the optimal solution!

Given n items with values $v_1, v_2, \ldots, v_n$ and weights $w_1, w_2, \ldots, w_n$ along with a knapsack capacity W, compute a subset of items with almost the maximum total value satisfying the capacity constraint.

Your answer will be considered correct if the total value in your knapsack is at least 1 times the total value in an optimal knapsack.

Input

The first line of input contains two space-separated integers n and W. Each of the following n lines has two space-separated integers v_i, w_i indicating the value and weight of the i-th item.

- $1 \leq n \leq 40$
- $1 \leq W, v_i, w_i \leq 10^9$

Output

Please output an integer indicating the maximum total value in an optimal knapsack.

Subtask 1 (45 %)

- $n \leq 20$

Subtask 2 (55 %)

- no other constraints.

Sample Input

```
4 5
3 2
4 4
5 3
2 1
```

Sample Output

```
8
```

Hint

- 2^{40} is pretty large, but $2^{20} = 1048576$ is a reasonable number :).
- Meet-in-the-middle?

Problem 3 - Candy (Programming) (20 points)

Problem Description

Joe is a good teacher of a class of n students. Christmas is coming, so he wants to buy some gifts for students. Since he is living in abject poverty, the only gifts he can afford are n candies. There are three types of candies (A, B, C) in the grocery store, and there is a discount if someone buys Candy A more than an amount.

Eventually each student will get a candy of any type. However, some students are friends and they don't want to have the same type of candy. Thus, Joe is going to find out a perfect candy distribution so that he can buy as much Candy A as possible.

Input

The first line contains two integers n, m indicating the number of students and the number of relationships. Each of the following m lines contains two integers x, y indicating student x and student y are friends.

- $2 \leq n \leq 20$
- $2 \leq m \leq n * (n-1)/2$
- $1 \leq x_i < y_i \leq n$

Output

Print an integer in one line indicating the maximum number of Candy A. If it is impossible to distribute the candies, output -1 instead.

Subtask 1 (30 %)

- $n \leq 10$

Subtask 2 (70 %)

- no other constraints.

Sample Input 1

```
4 3
1 2
1 3
1 4
```

Sample Output 1

```
3
```

Sample Input 2

```
7 7
1 2
1 3
1 4
4 6
3 4
3 5
5 7
```

Sample Output 2

```
4
```

Problem 4 (Hand-Written) (15 points)

Please approximately solve the *0/1 Knapsack* problem. Given n items with values $v_1, v_2, \ldots, v_n$ and weights $w_1, w_2, \ldots, w_n$ along with a knapsack capacity W, compute a subset of items with almost the maximum total value satisfying the capacity constraint.

(a) (4 points)

Prove that the following algorithm admits 2-approximation algorithm.

Place items into your knapsack until overflow in the order of decreasing value-to-weight ratio. Then compare the knapsack with the item with maximum value and pick the better choice.

Solution

Take items in the order of decreasing value-to-weight ratio is an obvious solution for fractional knapsack problem (being able to take a portion of an item). The optimal value in the fractional case is never smaller than the value in the 0/1 case given the same input.

By discarding the last fractional item in the fractional greedy solution, the total value is either halved or not. If the latter case happens, the discarded item must have value bigger than half the optimal value in the fractional case. But we compare all single item sets with the truncated greedy solution, both cases are handled and the solution acquired has value at least half the fractional optimal value.

(b) (3 points)

Let S^* be the subset in the optimal knapsack and V^* be the total value of S^*.

Consider the DP formulation $DP(i, V)$ being the lightest knapsack with total value at least V. Derive an algorithm solving the *0/1 Knapsack* problem in $O(nV^*)$-time via the previous DP formulation.

Solution

Let's write down the relation:

$$\begin{aligned}
&\mathbf{DP}(0,0) = 0 \\
&\mathbf{DP}(0,x) = \infty, \text{if } x \neq 0 \\
&\mathbf{DP}(i,v) = \min\{\mathbf{DP}(i-1,v), \mathbf{DP}(i,v+1), \mathbf{DP}(i-1,v-v_i)+w_i\}
\end{aligned}$$

A $(n+1)$-by-$2V_a$ table is initialized and filled with ∞. V_a is the value of the 2-approximated solution obtained by 4.(a). The table is filled row-wisely. Then, the row is filled backward. The time and space spent here is $O(nV_a) = O(nV^*)$.

(c) (2 points)

Let K be some positive magic integer. Some magician modifies values of all items by setting v_i with $\lfloor v_i/K \rfloor$. Derive an algorithm solving the modified knapsack problem in $O(\frac{nV^*}{K})$-time where V^* is still the optimal total value of the unmodified problem.

Solution

Let $\hat{V}$ be the optimal value of the modified problem. By 4.(b), we immediately have a $O(n\hat{V})$-time solution for the problem. To achieve the desired time complexity, we want to prove that $\hat{V} = O(\frac{V^*}{K})$.

$\hat{S}$ is a set of items that achieve optimal in the modified solution. We have:

$$\hat{V} = \sum_{i \in \hat{S}} \lfloor v_i/K \rfloor \leq \sum_{i \in \hat{S}} v_i/K \leq V^*/K \tag{1}$$

The last inequality comes from the fact that $\hat{S}$ has valid total weight and hence a valid set of items in the original problem.

(d) (6 points)

Let $\hat{V}$ be the optimal total value of the modified problem. Prove that $V^* - nK \leq K\hat{V} \leq V^*$.

Solution

The second inequality is proved in the solution of 4.(c). Now we focus on the first inequality.

Similarly, S^* is the optimal set of items for the original problem. Because weight of items are the same in both problems, S^* is also a valid solution to the modified problem.

$$\hat{V} \geq \sum_{i \in S^*} \lfloor v_i/K \rfloor \geq \sum_{i \in S^*} (v_i/K - 1) \geq \sum_{i \in S^*} v_i/K - n \tag{2}$$

Hence,

$$K\hat{V} \geq \sum_{i \in S^*} v_i - nK = V^* - nK \tag{3}$$

which concludes the proof.

Problem 5 (Hand-Written) (15 points)

Riot Games, the developer of League of Legends (LoL), is going to publicize a new game called "LoL 2". "LoL 2" is a game for two teams to compete against each other. To increase game variability, there are variable n players in each game of "LoL 2" (n is an integer larger than 1.). The chief developer, Eddy, thinks this setting will make "LoL 2" be more interesting than LoL. However, it is quite difficult to have a reasonable way to divide these n players into two teams with approximately equal strength.

Formally, each player i has a rating r_i which is an integer and ≥ 0, and the strength of a team, T, is defined as the sum of each player's rating in the same team, $S_T = \sum_{i \in T} r_i$. Given n players with corresponding ratings, $\{r_1, r_2, \ldots, r_n\}$, and these players must be divided into two teams, A and B, Eddy wants to minimize

$$\frac{\max(S_A, S_B)}{\min(S_A, S_B)} \tag{4}$$

Could you help him?

(a) (3 points)

Exact Set Cover problem

Given a collection of $\mathcal{S}$ of subsets of a set X, an **exact set cover** is a subcollection $\mathcal{S}^*$ of $\mathcal{S}$ such that each element in X is contained in *exactly one* subset in S^*. The problem is whether there exists an exact set cover.

Assuming Exact Set Cover problem is NP-hard, prove the following problem is NP-hard:

Given n players with corresponding ratings, $\{r_1, r_2, \ldots, r_n\}$ where r_i is an integer and ≥ 0, and a target strength, K, whether it is possible to form a team from these players, strength of which is equal to K. (sum of a subset of $\{r_1, r_2, \ldots, r_n\}$ equal to K)

Solution

The problem is actually the subset sum problem mentioned in Problem 6. Therefore, here we show how to reduce the exact set cover problem to the subset sum problem.

Assume that the cardinality of X is M and there are N subsets S_i in S, for each S_i, we make a player with the rating $r_i = \sum_{k \in S_i} 2^{k \times N}$ and we set the target $K = \sum_{k \in [0,M)} 2^{k \times N}$. Then, if we can find a subset of $\{r_1, r_2, \ldots, r_n\}$ sum of which is equal to K, there exists an exact set cover.

Note that if there exists a solution in such a subset sum problem, for each bit of K expressed in binary numeral system, there must exists exactly one r_i corresponding bit of which is also 1. This can be proved by contradiction, and the idea behind the proof is basically based on that we separate each 1 bit with a "distance" of N bits in the formulation $r_i = \sum_{k \in S_i} 2^{k \times N}$ and $K = \sum_{k \in [0,M)} 2^{k \times N}$, so that for N bits in the same position, they could only affect bits within $\log N$ distance.

(b) (4 points)

Assuming the problem in (a) is NP-hard. Given a number K, prove that to determine whether Equation (4) can be less than K or not is NP-hard.

Solution

We can reduce the subset sum problem to this problem with $K = 1+\epsilon$ where $\epsilon \to 0$. Let $S = \sum_i r_i$ and the target sum K', we add two more players with ratings $2S - K'$ and $S + K'$. The total sum will be $S + (2S - K') + (S + K') = 4S$ so those two players, $(2S - K')$ and $(S + K')$, cannot be in the same team $(3S > \frac{4S}{2} = 2S)$. Therefore, if there exists a solution with Equation (4) $< 1 + \epsilon$, we have a subset of $\{r_1, r_2, \ldots, r_n\}$ sum of which is equal to K' since $2S - (2S - K') = K'$.

(c) (6 points)

Consider the following algorithm and prove that it is an $O(1)$-approximation algorithm:

For $i = 1$ to n, assign the player i to the team which currently has lower strength.

Solution

Assume S_A and S_B are the output of the algorithm, and $S_A \geq S_B$ ($\frac{\max(S_A,S_B)}{\min(S_A,S_B)} = \frac{S_A}{S_B}$), we know that

1. $S_A \leq S_B + \max r_i$
2. $\frac{S_A}{S_B} \geq OPT \geq 1$

If $\frac{\max r_i}{S_A+S_B} \leq \frac{1}{2}$ $(\max r_i \leq 0.5(S_A + S_B))$,

$$\frac{S_A}{S_B} \leq \frac{S_B + \max r_i}{S_B} \leq 1 + \frac{0.5(S_A + S_B)}{S_B} \leq 1.5 + 0.5\frac{S_A}{S_B} \Rightarrow \frac{S_A}{S_B} \leq 3 \leq 3OPT \tag{5}$$

If $\frac{\max r_i}{S_A+S_B} > \frac{1}{2}$, $OPT = \frac{\max r_i}{S_A+S_B-\max r_i} \Rightarrow \max r_i = \frac{OPT}{1+OPT}(S_A + S_B)$

$$\frac{S_A}{S_B} \leq \frac{S_B + \max r_i}{S_B} \leq 1+\frac{\frac{OPT}{1+OPT}(S_A + S_B)}{S_B} \leq \frac{1 + 2OPT}{1 + OPT} + \frac{OPT}{1 + OPT}\frac{S_A}{S_B} \Rightarrow \frac{S_A}{S_B} \leq 1+2OPT \leq 3OPT \tag{6}$$

$\Rightarrow$ It is a 3-approximation algorithm.

(d) (2 points)

Provide a matching lower bound example for your proof which means if your proof shows that it is a k-approximation algorithm, then provide a case which the algorithm will find a ratio that is exactly k times larger than the optimal.

Solution

$\{r_1, r_2, r_3\} = \{1, 1, 2\}$ is an example for the 3-approximation algorithm.

Problem 6 (Hand-Written) (15 points)

Here we present some interesting problems.

Subset Sum

Given n non-negative integers $a_1, a_2, \ldots, a_n$ and an positive integer W, the *Subset Sum* problems seeks whether there is a subset whose sum equals to W. This problem is known to be NP-complete.

Partition

Given n non-negative integers $a_1, a_2, \ldots, a_n$ with sum being U, the *Partition* problems seeks whether there is a subset whose sum equals to $\frac{U}{2}$.

Bin Packing

Given n balls with weights $a_1, a_2, \ldots, a_n$ at most 1 kilogram, the *Bin Packing* problems seeks to partition balls into minimum number of bins of weight limit 1 kilogram.

(a) (0 point)

Reduce *Subset Sum* to *Partition* in polynomial time.

Solution

Use conclusions from problem 5.(a) and (b).

(b) (3 points)

Reduce *Partition* to *Bin Packing* in polynomial time.

Solution

Given the input for *Partition* problem. WLOG, we can assume $a_i \leq U/2$ for every i. Let $b_i = a_i/(U/2)$. If $b_1, b_2, ..., b_n$ can be packed in 2 bins, both bins must contain 1kg. of items since $\sum b_i = \sum a_i/(U/2) = 2$. And we have a solution for the original *Partition* problem. And if $a_1, a_2, \ldots, a_n$ can be partitioned into 2 sets each of total sum $\frac{U}{2}$, their corresponding b_is can be packed in 2 bins as well.

(c) (3 points)

If $\mathbf{NP} \neq \mathbf{P}$, show that there is no polynomial time $(\frac{3}{2} - \epsilon)$-approximation for *Bin Packing* problem for any $\epsilon > 0$.

Hint: Consider the reduction in (b).

Solution

If we have such an algorithm, we can solve the NP-hard *Partition* problem in polynomial time.

Given a problem instance of *Partition*, we reduce it to a *Bin Packing* instance and ask if the answer is larger than 2. If the answer is 2, the algorithm would output a integer x such that $x \leq (\frac{3}{2} - \epsilon)2 = 3 - 2\epsilon$. And clearly x must be 2. Otherwise, the algorithm outputs an answer bigger than 2. Hence, it's a correct and fast algorithm for the *Partition* problem.

(d) (3 points)

Let's try to solve *Bin Packing* problem in polynomial time. Suppose all balls weigh at least $\frac{1}{3}$ kg and there are only 5 different weights.

Suppose there is at most T possibilities for the content of a single bin, please prove that $T \leq 65$.

Solution

Because of the weight lower bound, there are at most 3 balls in a single bin. So the number of types of valid content in a single bin is at most $\sum_{i=0}^{3} \binom{5}{i} = 26 \leq 65$.

(e) (3 points)

Given k indistinguishable bins, prove that there is $O(k^{65})$ possibilities for the whole content inside these bins.

Solution

From (d), there is at most 65 possibilities for a single bin. So the possibilities for k indistinguishable bins is bounded by $\binom{k-1+65}{65} = O(k^{65})$.

(f) (3 points)

Derive and justify a **polynomial time** algorithm for the *Bin Packing* problem under the above constraints.

Solution

First, we enumerate all possible number of bins from 1 (use only 1 bin to pack n balls) to n (1 bin per ball). And for each k, we check all possible packing content in $O(k^{65})$-time. Overall, the time of this algorithm is $O(n^{66})$.