CS 630, Fall 2024, Homework 2

Yifei Bao

September 23, 2024

Problem 1 ***Price contest with a blackbox algorithm***

Algorithm 1: PriceContest($[a_1, a_2, ..., a_n], B$)

```
1 V ← [ ]/* Initialize an empty list of selected objects                    */
2 T ← B/* Initialize the remaining budget with the initial budget           */
3 for i = 1...n do
4     if a_i ≤ T AND CanSolve([a_i, ..., a_n], T) then
5         V.append(a_i);
6         T ← T − a_i;
7         if T == 0 then
8             return V;
9 return "not possible";
```

Algorithm Description

The algorithm takes a list of objects $[a_1, a_2, ..., a_n]$ and a budget B as input. The output is a list of objects, total value of which is exactly B.

The algorithm uses empty list V to store the selected objects and T to store the remaining budget from B.

For each object a_i, if the object's value is no more than the remaining budget and the blackbox function CanSolve finds that there exists a subset of remaining objects, the sum of which equals to remaining budget, then the object is selected and added to V. The object's value is subtracted from the remaining budget.

The algorithm returns the selected objects if there is exactly no remaining budget. Otherwise, it returns "not possible".

Correctness Proof

In each step, the algorithm uses the blackbox function CanSolve to check if there exists a subset of remaining objects that sums to the remaining budget. If the function returns true, the object is selected and the remaining budget is updated. CanSolve ensures that the selected objects can sum to the remaining budget, so after iterating through all objects, the selected objects will sum to the initial budget B. Thus the algorithm is correct.

Time Complexity
In the worst case, the algorithm will iterate through all n objects. For each object, it will call the blackbox function CanSolve, which takes $O(C)$ time. So the total running time is $O(nC)$.

Space Complexity
The space complexity is $O(n)$ for storing the list of selected objects.

Problem 2 *Vaccine testing*

Algorithm 2: VaccineTesting(D, T)

```
1  n ← len(D);
2  P ← [p_0, p_2, ..., p_{n-1}] /* Initialize the person list, each p is a person */
3  S ← [ ] /* Initialize the list of subsets */
4  for i = 0 ... n − 1 do
5      S_i ← [ ];
6      for j = 0 ... n − 1 do
7          if D[i][j] ≤ T then
8              S_i.append(p_j)/* For each person p_i, add all persons within distance
                   T to S_i as representative */
9      S.append(S_i)/* Add the representative set of person p_i to S */
10 minSetCover ← SCSolver(P, S)/* Get minimum set cover of P */
11 representativeSet ← GetPerson(minSetCover)/* Get persons of corresponding
       sets, i.e.  for sets S_1, S_3, get p_1, p_3 */
12 return representativeSet;
```

Algorithm Description

The algorithm takes distance matrix D and max distance value T as input.

The purpose of the algorithm is to select a minimum set of individuals such that every person in the population is within distance T of at least one person in the set.

In this algorithm, for each person p_i, we find all persons within distance T to p_i and add them to set S_i. It basically means that p_i can represent all persons in S_i because they are within distance T to p_i.

Then we get the minimum set cover of P in S using the blackbox function SCSolver. And it means that the minimum set of persons in P can cover all persons in the population.

Finally we get the persons of representing each set in the minimum set cover and return them as the representative set.

Correctness Proof

1. Every person in the population is represented. Constructing S_i of each person p_i ensures that every person is included in at least one set in S. The SCSolver ensures that the minimum cover sets of P covers all persons in the population. Thus every person is represented by at least one of the selected set.

2. The number of trial individuals is minimized. The SCSolver ensures that the minimal number of persons in P is selected to cover all persons in the population.

Time Complexity
Total running time: $O(n^2 + C)$
$1, 2, 3$ $O(n)$ initializing n,P,S
5 $O(n)$ initializing S_i for each person
$7, 8$ $O(n^2)$ checking distance for each pair of persons and adding to S_i
9 $O(n)$ adding S_i to S
10 $O(C)$ calling SCSolver
11 $O(n)$ getting persons of corresponding sets

Space Complexity
The space complexity of the algorithm is $O(n^2)$ for storing the sets S and $O(n)$ for storing the person list P. The total space complexity is $O(n^2 + n) = O(n^2)$.