EECS 336

Litong Wang

Homework 5

May 9, 2016

Problem 1

Solution:

We should go as far as possible before we need to refill the tank. Let's assume a_i to be the gas station which has distance d_i from Chicago, A to be the set contains gas station where we should stop.

Algorithm:

```
Data: $d_i, m$
Result: $A$
$A = \emptyset$ ;
$laststop = 0$ ;
for $i = 1; i \leq n; i++$ do
    if $d_i - laststop > m$ then
        $A = A \cup a_{i-1}$ ;
        $laststop = d_{i-1}$ ;
    end
end
return $A$ ;
```

Problem 2

Solution:

Problem 3

Solution:

Problem 4

Solution:

We should sort sets A and B into monotonically increasing order. Then we can use this order to achieve the maximum payoff $\Pi_{i=1}^{n} a_i^{b_i}$.

Proof:

Consider any two indices such that $i < j$, we can have two items, $a_i^{b_i}$ and $a_j^{b_j}$. We need to prove that the payoff which includes $a_i^{b_i}$ and $a_j^{b_j}$ is greater than payoff that includes $a_i^{b_j}$ and

$a_j^{b_i}$. We only need to prove that $a_i^{b_i}a_j^{b_j} \geq a_i^{b_j}a_j^{b_i}$. Because A and B are sorted into monotonically increasing order, $a_i \leq a_j \; and \; b_i \leq b_j$. We divide $a_i^{b_i}a_j^{b_j}$ and $a_i^{b_j}a_j^{b_i}$ by $a_j^{b_i}a_i^{b_i}$, then we can have $a_j^{b_j-b_i}$ and $a_i^{b_j-b_i}$. Because $b_j - b_i \geq 0$ and $a_j \geq a_i$, we can have $a_j^{b_j-b_i} \geq a_i^{b_j-b_i}$.

$$a_j^{b_j-b_i} \geq a_i^{b_j-b_i} \Leftrightarrow a_i^{b_i}a_j^{b_j} \geq a_i^{b_j}a_j^{b_i}$$

The running time for sorting is $\mathcal{O}(n \log n)$ and the time for computing payoff is $\mathcal{O}(n)$. Hence, the running time for this algorithm is $\mathcal{O}(n \log n)$.

Problem 5

Solution:

Quarter = 25 cents, dime = 10 cents, nickel = 5 cents and penny = 1 cent.
The coin changing problem has optimal substructure. Assume we have an optimal solution for making change for n cents, we use a coin whose value is v and we use k coins in this solution. This optimal solution must contain an optimal solution for problem of making change for n-v cents. k-1 coins are used in the optimal solution to n-v cents problem used within optimal solution to the n cents problem. If we can find a solution that uses coins fewer than k-1 coins for n-v cents problem, then we can cut and paste this solution to the n cents problem. In this case, we can find a solution for n cents problem that uses coins fewer than k. This contradicts the optimality of the solution.

a) A greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies:
Let's denote $count_{quarter}$ to be the number of quarters used in the solution. $count_{quarter} = \left\lfloor \frac{n}{25} \right\rfloor$, and the coins remain are $n_q = n \mod 25$.
Let's denote $count_{dime}$ to be the number of quarters used in the solution. $count_{dime} = \left\lfloor \frac{n_q}{10} \right\rfloor$, and the coins remain are $n_d = n_q \mod 10$.
Let's denote $count_{nickel}$ to be the number of quarters used in the solution. $count_{nickel} = \left\lfloor \frac{n_d}{5} \right\rfloor$, and the coins remain are $n_p = n_n \mod 5$. $count_{penny} = n_p$
Correctness:
To prove this algorithm yields an optimal solution, we need first shows that the geedy-choice property holds. The geedy-choice property is that an optimal solution to making change for n cents includes a coin of value v, which is the largest value such that $v \leq n$. Consider an optimal solution, if this optimal solution contains v, then we can finish our proving. If the optimal solution doesn't contain value v, then we need to take four cases into consideration.

- If $1 \leq n < 5$, then we have $v = 1$. Ths solution can only consist of pennies, and so it must contain the greedy choice.
- If $5 \leq n < 10$, then we have $v = 5$. By supposition, the optimal solution doesn't contain

a nickel. We can replace five pennies by a nickel, in this case, we can get a solution with four coins fewer.

• If $10 \leq n < 25$, then we have $v = 10$. By supposition, the optimal solution doesn't contain a dime. We can replace pennies and nickels which adds up to 10 cents by a dime, in this case, we can get a solution with fewer coins.

• If $25 \leq n$, then we have $v = 25$. By supposition, the optimal solution doesn't contain a quarter. The solution can only consist of pennies, nickels and dimes. If we have three dimes, we can replace them with a quarter and a nickel. In this case, we can get a solution with one fewer coin. If pennies, nickels and dimes adds up to 25 cents, we can replace them with a quarter. In this way, we can get a solution with fewer coins.

Thus, we can see that there is always an optimal solution that includes the geedy choice. Therefore, we can combine the greedy choice with an optimal solution to the subproblem to produce the optimal solution to the original problem.

Time complexity:

We choose one coin at a time and compute the subproblem recursively, the running time is $\Theta(k)$, k is the number of coins used in an optimal solution. Because $k \leq n$, the running time is $\mathcal{O}(n)$. In each computation, the running time is $\mathcal{O}(1)$. Hence, the running time of this algorithm is $\mathcal{O}(n)$.

b) Show that the greedy algorithm always yields an optimal solution.

The greedy algorithm for making change for n cents is to find the denomination c^j such that $j = \max\{0 \leq i \leq k : c^i \leq n\}$. And then we compute the subproblem of making change for $n - c^j$. Correctness:

If in the optimal solution, we use a_i coins of denomination c^i to make change for n cents, then $a_i < c$. If $a_i \geq c$, we can replace c c^i with a c^{i+1}, then we can reduce the total number of coins by $c - 1$.

If we don't use a greedy algorithm to solve the problem of making change for n cents, we can get a solution which uses denominations $c^0, c^1, \cdots, c^{j-1}$ and $\sum_{i=0}^{j-1} a_i c^i = n$, also $n \geq c^j$. Because $n \geq c^j$, $\sum_{i=0}^{j-1} a_i c^i \geq c^j$. Let's assume that the non-greedy algorithm is an optimal solution, then we can get $a_i \leq c - 1$ for $i = 0, 1, \cdots, j-1$. Then we can get the following conclusion:

$$\begin{aligned}\sum_{i=0}^{j-1} a_i c^i &\leq \sum_{i=0}^{j-1}(c-1)a^i \\ &= (c-1)\sum_{i=0}^{j-1} a^i \\ &= (c-1)\frac{1-c^j}{1-c} \\ &= c^j - 1 \\ &< c^j\end{aligned}$$