Algorithm Homework 2

Jingwei Zhang 201528013229095

2015-10-13

1 Problem 1

1.1 Optimal Substructure

1.2 Algorithm

1.3 Correctness

1.4 Complexity

2 Problem 2

2.1 Optimal Substructure

2.2 Algorithm

2.3 Correctness

2.4 Complexity

3 Problem 5

3.1 Optimal Substructure

3.2 Algorithm

3.3 Correctness

3.4 Complexity

4 Problem 6

4.1 Algorithm

4.2 C++ Code

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <algorithm>
#include <vector>

using namespace std;

void fill_left(vector<int> &p, const vector<int> &d){
```

```
    p.resize(d.size());
    if(p.size() > 0){
        // Firstly, p[i] represents the max profile you can get
        // when you sell the stock in day i

        // price_min means the minimum price in [0,i]
        // when i iterates in array d
        int price_min =  d[0];
        p[0] = 0;

        for(int i = 1; i < d.size();i++){
            price_min = min(price_min, d[i]);
            p[i] = d[i] - price_min;
        }

        // Now compute the max profile you can get during [0,i]
        // Store it in p[i]

        // profile_max maintains the max in p[0,i]
        int profile_max = 0;
        for(int i = 0;i < p.size();i++){
            profile_max = max(profile_max, p[i]);
            p[i] = profile_max;
        }
    }
}

void fill_right(vector<int> &p, const vector<int> &d){
    p.resize(d.size());
    if(p.size() > 0){
        // Firstly, p[i] represents the max profile you can get
        // if you buy the stock in day i

        // price_min means the maximum price in [0,i]
        // when i iterates reversely in array d
        int price_max = p[p.size()-1];
        p[p.size()-1] = 0;

        for(int i = p.size()-1;i >= 0;i--){
            price_max = max(price_max, d[i]);
            p[i] = price_max - d[i];
        }

        // Now compute the max profile you can get during [i,end]
        // Store it in p[i]
        int profile_max = p[p.size()-1];
        for(int i = p.size()-1;i >= 0;i--){
            profile_max = max(profile_max, p[i]);
            p[i] = profile_max;
        }

    }
}

int main()
{
```

```
    freopen("stocks.in","r",stdin);
    //freopen(".out","w",stdout);
    vector<int> d;
    int t;
    while(cin>>t){
        d.push_back(t);
    }

    // pre[i] stores the max profit you get during day [0, i]
    // in a single transaction
    vector<int> left;


    // last[i] stores the max profit you get during day from i to last
    // in a single transaction
    vector<int> right;

    fill_left(left,d);
    fill_right(right,d);

    int sum_max = 0;
    for(int i = 0;i < left.size();i++){
        sum_max = max(sum_max, left[i] + right[i]);
    }
    cout<<sum_max<<endl;

    return 0;
}
```