CS 477: Homework #5

Due on November 3rd, 2016 at 2:30pm

Monica Nicolescu

Matthew J. Berger

Problem 1

1.) (U & G Required)[100 points]

Suppose you are consulting for a company that manufactures PC equipment and ships it to distributors all over the country. For each of the n next weeks, they have a projected supply si of equipment (measured in pounds), which has to be shipped by an air freight carrier. Each weeks supply can be carried by one of two air freight companies, A or B.

- Company A charges a fixed rate r per pound, so it costs $r * s_i$ to ship a weeks supply (s_i)
- Company B makes contracts for a fixed amount c per week, independent of the weight. However, contracts with company B must be made in blocks of four consecutive weeks at a time.

A schedule, for the PC company, is a choice of air freight company (A or B) for each of the n weeks with the restriction that company B, whenever it is chosen, must be chosen for blocks of four contiguous weeks at a time. The cost of the schedule is the total amount paid to companies A and B, according to the description above.

You are asked to give a polynomial time algorithm that takes a sequence of supply values $s_1, s_2, \ldots, s_n$ and returns a schedule of minimum cost. In order to achieve this, you need to answer the following questions:

a.) [20 points] Determine and **prove** the optimal substructure of the problem and write a recursive formula of an optimal solution (i.e., define the variable that you wish to optimize and explain how a solution to computing it can be obtained from solutions to subproblems).
Submit: the recursive formula, along with definitions and explanations on what is computed.

Solution

We'll call the algorithm $MINCOST(i)$. In this case, the algorithm computes the lowest cost possible to ship the the PC equipment to distributors for the first i weeks. We'll also declare an algorithm called $OPTIMIZE(i, j)$ to find the company for the j-th week that would achieve $MINCOST(i)$. The best schedule for the first i weeks will either be acquired through choosing company A for the i-th week (a single week) or by choosing company B for the previous 3 weeks as well as the i-th one (weeks $i, i-1, i-2$, and $i-3$).

The optimal substructure of this problem can be represented by the equation below:

(1) $$MINCOST(i) = min\{MINCOST(i-1) + r * s_i,\ MINCOST(i-4) + 4c\} \text{ for } i \geq 4$$

However for $i < 4$, we must choose company A for each week:

(2) $$MINCOST(0) = 0$$

(3) $$MINCOST(i) = MINCOST(i-1) + r * s_i$$

b.) [30 points] Write an algorithm that computes an optimal solution to this problem, based on the recurrence above. Implement your algorithm in C/C++ and run it on the following values:

- $r = 1$
- $c = 10$
- the sequence of s_i values: 11, 9, 9, 12, 12, 12, 12, 9, 9, 11

Submit:

- A printed version of the algorithm (name your algorithm schedule.c or schedule.cpp)
- A printout of the table that contains the solutions to the subproblems, run on the values given above (print the entire table!)

Solution

```cpp
#include <algorithm>

#define NUM_WEEKS 10
#define COMPANY_A 0
#define COMPANY_B 1

int main()
{
    int r = 1;
    int s[NUM_WEEKS] = { 11, 9, 9, 12, 12, 12, 12, 9, 9, 11 };
    int c = 1;
    int minCost[NUM_WEEKS] = {0};

    int opt[NUM_WEEKS][NUM_WEEKS] = {{0}};

    for(int i = 1; i < NUM_WEEKS; i++)
    {
        if(i < 4)
        {
            minCost[i] = minCost[i-1] + (r * s[i]);

            for(int j = 1; j < i-1; j++)
            {
                opt[i][j] = opt[i-1][j];
            }
            opt[i][i] = COMPANY_A;
        }
        else
        {

            int costA = minCost[i-1] + (r * s[i]);
            int costB = minCost[i-4] + (4 * c);
            minCost[i] = std::min(costA,costB);

            if(costA < costB)
            {
                for(int j = 1; j < i-1; j++)
                {
                    opt[i][j] = opt[i-1][j];
                }
                opt[i][i] = COMPANY_A;
            }
            else
            {
                for(int j = 1; j < i-4; j++)
                {
                    opt[i][j] = opt[i-4][j];
                }

                for(int x = 3; x >= 0; x--)
                {
                    opt[i][i-x] = COMPANY_B;
                }
            }
        }
    }

    for(int i = 0; i < NUM_WEEKS; i++)
    {
        std::string choice = (opt[i][i] == COMPANY_A) ? "Company A" : "Company B"↩
            ;
```

```
61          std::cout << "Week " << i << ": " << choice << std::endl;
62      }
63
64      return 0;
65  }
```

c.) [20 points] Update the algorithm you developed at point (b) to enable the reconstruction of the optimal solution, i.e., which company was used in an optimal solution for shipping. (Hint: use an auxiliary table like we did in the examples in class.) Include these updates in your algorithm implementation from point (b).
Submit:

- A printed version of the algorithm (name your algorithm schedule_1.c or schedule_1.cpp).
- A printout of the values that you obtain in the table containing the additional information needed to reconstruct the optimal solution, run on the values given above (print the entire table!)

d.) [30 points] Using the additional information computed at point (c), write an algorithm that outputs which company was used for shipping in the optimal schedule. Implement this algorithm in C/C++.
Submit:

- A printed version of the algorithm (name your algorithm schedule_2.c or schedule_2.cpp).
- A printout of the **solution** to the problem, i.e., the optimal schedule. (e.g., A, A, B, A, B)