CS29002 Algorithms Laboratory

Assignment No: 9

Last date of submission: 05–October–2016

Program 1

In the class, you have seen algorithmic solutions of the fractional and $0,1$ knapsack problems. Here, you handle a variant of the knapsack problem.

You have a knapsack of weight capacity B. There are n types of objects. There is an infinite supply of objects of each type, that is, you can pack multiple objects of the same type in the knapsack. Each object in a given type has the same weight and the same cost (profit)—call these w_i and c_i for the i-th type. Your task is to pack objects in your knapsack such that the capacity constraint of the knapsack is not exceeded (that is, the total weight of the objects packed is $\leqslant B$), and the total cost of the objects packed is as large as possible.

Devise a dynamic-programming algorithm to solve this variant of the knapsack problem. Write a program *knapsack.c* to implement your algorithm. The program first reads n, B, $w_1, w_2, \ldots, w_n$, and $c_1, c_2, \ldots, c_n$ from the user. The program then calculates and prints the optimal cost for these input values. It should also print the counts of the chosen objects of different types.

Sample output

```
+++ n = 5
+++ B = 1000
+++ Weights : 21 23 24 22 24
+++ Costs   : 23 25 24 24 22

+++ Maximum cost = 1094
+++ Weight = 1000
+++ Counts  : 40  6  0  1  0
```

Program 2

In the twelfth century, a research team in the Indian Institute of Alchemists was working with four liquids a, b, c, d. For each pair (s,t) of these liquids, they followed a mixing procedure. They took some quantity of Liquid s in a special pot, and added the same quantity of Liquid t to the pot. Heating the mixture gave Liquid r, where $r \in \{a,b,c,d\}$. The mixing procedure was not symmetric, that is, the pairs (s,t) and (t,s) might result in different liquids. Suppose that the result of mixing Liquid s to itself is again Liquid s. The research team knew the results of mixing all the sixteen possible liquid pairs.

A correspondent from the Ras Yang province of China gave a string $\alpha \in \{a,b,c,d\}^*$ as a potentially useful way of preparing one of the four liquids. The history of making this liquid was believed to give a formula for manufacturing gold. The string was supposed to store this history. Unfortunately, the string did not make it clear the exact way of mixing the liquids. For example, the string $\alpha = dacb$ may mean $(d(a(cb)))$, $(d((ac)b))$, $((da)(cb))$, $((d(ac))b)$ or $(((da)c)b)$, depending upon how it is explicitly parenthesized. The leader of the research team, Uparasayanacharya, asked his team to investigate which liquids could be obtained by different ways of parenthesizing the string α, and each in how many ways.

Solve Uparasayanacharya's problem in two phases.

- Write a function $isrealizable(s, \alpha, T)$ that, given an $s \in \{a,b,c,d\}$ and the string α, decides and prints whether some parenthesization of α gives the final result s. Here, T is the 4×4 table storing the results of mixing pairs of liquid. You do not need to compute a parenthesization of α for realizing s. Only a binary decision needs to be output.

- Write a function *countpossibilities*(α, T) to determine in how many ways each $s \in \{a,b,c,d\}$ can be obtained by different ways of parenthesizing α, and to print all the four counts. It suffices only to compute the counts. You do not need to find out (or print) the results of all ways of parenthesizing α.

Both the two functions should use dynamic programming, run in $O(n^3)$ time, and use $O(n^2)$ memory, where n is the length of the string α.

Write a program *uparasayanacharya.c* that first asks the user to enter the results of mixing all pairs (s,t). The user then enters a non-empty string α. For each $s \in \{a,b,c,d\}$, call *isrealizable*() to print the decision about the realizability of s by α. Subsequently, call *countpossibilities*() in order to print the counts of different ways of realizing all $s \in \{a,b,c,d\}$ by α.

Sample outputs

```
+++ T
    a  b  b  c
    c  b  a  d
    b  a  c  b
    b  a  a  d
alpha = bbdc

+++ Checking realizability
    a is realizable
    b is not realizable
    c is realizable
    d is not realizable

+++ Counting possibilities
    a is realizable in 3 ways
    b is realizable in 0 ways
    c is realizable in 2 ways
    d is realizable in 0 ways
```

Here, the 4×4 mixture table is presented in the row-major format. For $\alpha = bbdc$, all parenthesizations are:

$$\begin{aligned}
(b(b(dc)) &= (b(ba)) = (bc) = a,\\
(b((bd)c) &= (b(dc)) = (ba) = c,\\
((bb)(dc)) &= \quad\quad\;\; (ba) = c,\\
((b(bd))c) &= ((bd)c) = (dc) = a,\\
(((bb)d)c) &= ((bd)c) = (dc) = a.
\end{aligned}$$

Now follows a bigger example ($|\alpha| = 12$).

```
+++ T
    a  a  b  a
    c  b  a  b
    a  d  c  d
    a  a  c  d
alpha = cdccdbcccddd

+++ Checking realizability
    a is realizable
    b is realizable
    c is not realizable
    d is realizable

+++ Counting possibilities
    a is realizable in 39533 ways
    b is realizable in 3462 ways
    c is realizable in 0 ways
    d is realizable in 15791 ways
```

Submit two C/C++ source files *knapsack.c* and *uparasayanacharya.c*. Do not use global/static variables.