CS29003 Algorithms Laboratory

Assignment No: 8

Last date of submission: 05–October–2017

A huge battleship has two decks (left and right) for stowing fighter airplanes. Each deck is of length L. There are n airplanes waiting in a queue to be loaded to the ship. The lengths of these airplanes are given in an array $A = (a_0, a_1, a_2, \ldots, a_{n-1})$. All lengths (the capacity L and the individual lengths a_i) are assumed to be positive integers. For each $i = 0, 1, 2, \ldots$ (in that sequence), you decide whether the i-th airplane will go to the left deck or the right deck. Your objective is to maximize the total number k of airplanes that can be stowed in the two decks without exceeding their respective stowing capacities L.

Part 1: Write a recursive function *exhsearch* to maximize k using exhaustive search. Let i be the number of airplanes loaded to the ship, and u, v the respective spaces (lengths) used in the two decks. Initially, $i = 0$, so $u = v = 0$. The function *exhsearch* takes i, u, v as input arguments (along with other necessary items like L, A, n). If all the airplanes are loaded ($i = n$), or if neither of the two decks can accommodate the next airplane (whose length is a_i), then i is returned. Otherwise, the function checks whether the left deck can accommodate the next airplane a_i. If so, it makes a recursive call by stowing that airplane in the left deck. An analogous conditional recursive call is made with the airplane a_i stowed in the right deck. The larger of the two returned values is returned.

Part 2: Implement a hash table T with chaining for storing the (i, u, v) triples defined in Part 1. The table should have size $s = nL$. Each chain should be stored as a linked list of (i, u, v) triples. Use the hash function

$$H(i, u, v) = 7i + 3u + 5v \pmod{s}.$$

Implement the following functions to manage T: *init* (build an initially empty hash table), *search* (check whether a triple (i, u, v) is already present in T), and *insert* (insert a triple (i, u, v) in T if not already present in T). This application does not require the deletion operation.

Part 3: Write a function *hashsearch* to find the maximum number k of airplanes that can be stowed in the ship. The function works very similarly as the function *exhsearch* of Part 1. The only exception is that if a recursive call leads to a triple $(i+1, u+a_i, v)$ or $(i+1, u, v+a_i)$ already present in the hash table T, then this recursive call is not made. This avoids multiple explorations from the same (i, u, v) triples, and brings down the running time from potentially exponential (in n) to $\Theta(nL)$, since the maximum number of triples (i, u, v) is about nL (this also justifies the choice $s = nL$ in Part 2).

The *main*() function

- Read L, n, and the individual lengths $a_0, a_1, a_2, \ldots, a_{n-1}$ from the user.
- Call *exhsearch*, and print the value of k returned. Also record and print the time taken by this call.
- Call *hashsearch*, and print the value of k returned. Also record and print the time taken by this call.

Sample output

```
    72 36
    4 1 4 3 4 2 3 9 2 9 7 7 3 9 4 9 3 4 7 3 7 9 3 4 4 1 1 1 5 9 9 8 7 3 8 1

+++ Exhaustive search
    k = 30
    Search time = 6.394342 sec

+++ Hash-based search
    Hash table of size 2592 initialized
    k = 30
    Search time = 0.000335 sec
```

Submit a single C/C++ source file. Do not use global/static variables.

Appendix: How to Measure Running Time

```
#include <time.h>

clock_t c1, c2;
double runtime;

c1 = clock();
/* Beginning of code whose running time you want to measure */
...
/* End of code whose running time you want to measure */
c2 = clock();

runtime = (double)(c2 - c1) / (double)CLOCKS_PER_SEC;
printf("Running time = %lf seconds\n", runtime);
```