Introduction to Computer Science
HW #4
Due: 2019/05/015

Homework Rules:

Hand-written homework can be handed in **before lecture starts**. Otherwise, you may contact the TA in advance and then bring the hardcopy to the TA in MD-631 (please send e-mail in advance).

As for the programming part, you need to upload it to CEIBA before the deadline(**2019/05/15 27am**) . The file you upload must be a **.zip** file that contains the following files:

README.txt

HW04_b05901XXX (a folder that contains all .cpp & .h as required),

1. Do not submit executable files (.exe) or objective files (.o, .obj). Files with names in wrong format will not be graded. You must **remove any system calls**, such as system ("pause"), in your code if any.
2. In README.txt, you need to describe which compiler you used in this homework and how to compile it (if it is in a "project" form).
3. In your .cpp files, we suggest you write comments as detailed as you can. If your code does not work properly, code with comments earns you more partial credits.

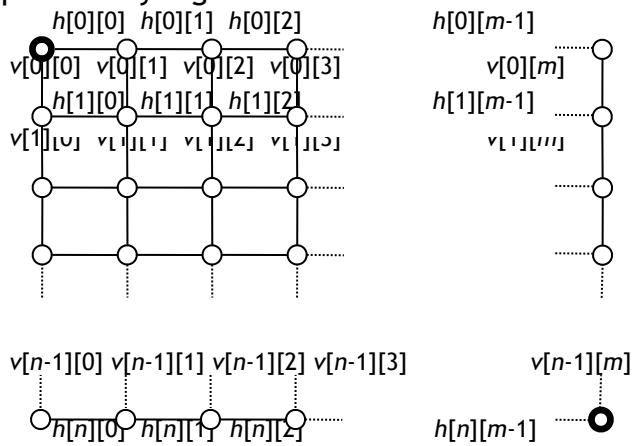
Chapter 6 Review Problems (5% each):

19,40,41,47,56

Programming Problem I (50%): Shortest Path

You may use either python or C/C++ for this homework; however, you need to deal with the "read" by yourself.

Consider a graph of *m* by *n* grids:



Use dynamic programming to find the shortest path from the upper-left node to the lower-right node with **moves in all four directions**. Costs of vertical edges are stored in an 2D array *v*[0..*n*-1][0..*m*]; costs of horizontal

edges are stored in another 2D array *h*[0..*n*][0..*m*-1]. These costs are **positive real-values**.

You can use readParameters() to read all parameters (*m*, *n*, *v*[][], and *h*[][]) from **input**. Remember to call **release()** when done. Check out hw4.cpp for more information.

Your program should print out 2 lines (on screen). The 1st line is the total **cost** of the shortest right-down path. The 2nd line is a string of characters of '**u**', '**d**', '**l**' or '**r**', standing for up, down, left or right respectively.

Programming Problem II (25%):

Write a **prolog** program to solve the maximum subarray problem: given an array a with length n, find

$$M = \max(L),\ L = \Sigma_{x=i}^{j} a[x]\ 1 \le i, j \le n$$

You need to write the function (save your code in **"msp.pl"**):

- ***maxsum*(*L*,*M*):** *solve the* maximum subarray problem and find maximum ***M*** for the given list ***L***.

Note that if given an array of all negative values, the maximum value is 0.

Bonus (5%)

Suppose that you have a set of n tasks with **integer** running time t_i in the range $[1, k]$. On a CPU with **2** cores, we try to balance the work load by partitioning tasks into two disjoint subsets S_1 and S_2 such that max($T(S_1)$, $T(S_2)$) is minimum, where $T(S)$ is the summation of the running time of the tasks in S.

$$T(S) = \sum_{i \in S} t_i$$

Design an efficient algorithm and write a program to solve the above problem. Your program should read a text file named "**input.txt**", which contains only one line listing n and t_i **in order** (you can find k in $\mathbf{O(n)}$). All numbers are separated by **space**. Your program should print out three lines. The 1st line is the loading difference: $|T(S_1) - T(S_2)|$. The 2nd and 3rd lines list the running times of the tasks in S_1 and S_2 respectively. All numbers should be separated by **space**.

Hint: pseudo-polynomial, integer range, DP

Example: https://www.geeksforgeeks.org/subset-sum-problem-dp-25/