CS575 Project 3

Due at 11:59pm on May 4
(Submit through blackboard)

1. [60%] Implement the following 0/1 knapsack algorithms.

   1) [10%] Implement the brute force method to solve the 0/1 knapsack problem. Show the final solution. Specifically, (a) print the total profit and weight; and (b) print the selected items.

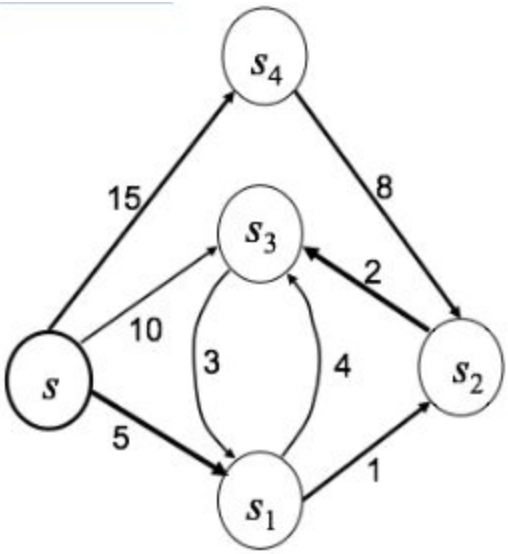
   2) [20%] Implement the four greedy algorithms discussed in class. For each algorithm, (a) print the total profit and weight; and (b) print the selected items together with their profits and weights.

   3) [30%] Implement the dynamic programming algorithm for 0/1 knapsack: (a) print the total profit and weight; and (b) print the selected items. Compare the result to the result you have got in 1). If you have implemented the dynamic programming algorithm correctly, the total profit achieved by this backtracking method must be equal to that achieved by the brute force method implemented in 1).

      Implement each algorithm as a separate program. Pass input file, knapsack.txt, to each program. Your knapsack.txt should include the number of elements in the first line, weights in the second line, profits in the third line, and knapsack capacity in the fourth line. Hence, it should look exactly as the following sample data (This just a sample. Your programs should work correctly for any input.):

      3
      5, 20, 10
      50, 140, 60
      30

2. [30%] Implement Dijkstra's algorithm to find the shortest path from *s* to all the other nodes in a directed graph. Use the following graph as a sample. (Your program should work correctly for any input though.)



3. [10%] 10% is reserved for good coding style, meaningful comments, and correctly following the submission guidelines below.

Submission Guidelines:

- Please submit a tar.gz file with the below naming convention. <user name>_project3.tar.gz
- Make sure you include all the source code files as mentioned below:

- Your makefile should be able to compile all the source code and generate .out files. make sure you makefile will not run the executable files.

- You should run and test your code in **remote.cs.binghamton.edu** and make sure it works there without any issues. You code will be evaluated in the same environment. If code does not run on remote server the TA will not run it on any other environments.

- Wrong directory structure and presence of unnecessary files will cost you points.

- Do not create unnecessary output files, header files, printf statements that are not asked to create.

- Please do not assume anything, email questions to the TA (avora4@binghamton.edu), and see him during his office hours for any required clarifications.

Important Policies:

- **All work must represent each individual student's own effort**. **You are not allowed to refer to any online or offline source.** If you show your code or any other part of your work to somebody else or copy or adapt somebody else's work found online or offline, you will get zero and be penalized per the Watson School **Academic Honesty Code** (http://www.binghamton.edu/watson/about/honesty-policy.pdf). To detect software plagiarism, everybody's code will be cross-checked using an automated tools. **On the first violation, you will receive zero point for this project, and have to fill in and sign the official form that will be kept by the Binghamton University. On the second violation, it will be reported to the Watson School Committee for Academic Honesty for an official hearing.**

- Your code will be compiled and executed. **If your code does not compile or produce any runtime errors such as segmentation faults or bus errors, you will get zero.** Before submitting your code, run it in a Linux machine (e.g., a remote machine) to remove any such errors. You can use "**valgrind**" (http://valgrind.org/) to debug memory errors, such as segmentation faults or bus errors. Note that **robust programming** is essential for developing high quality software. If you are curious about robust programming, check this out: http://nob.cs.ucdavis.edu/bishop/secprog/robust.html

- **You are supposed to figure out how to implement this project. The instructor and TA will be more than happy to explain fundamental algorithms covered in class; however, they will not teach you how to implement them or help you to make any kind of decision. Neither will they read or debug your code.** By the same token, the instructor and TA will not take a look at an emailed code. If you have questions about general C/C++ programming or Linux, use Google search, which will be much faster than asking the TA. (Don't copy from the Internet though.)

- **If myCourse says your submission is late, than it's late**. You can submit your code with late penalty 10% for each day upto 3 days max. After 3 days your submission won't be accept.