

## CS575 Programming Assignment 5

Due at 11:59:59PM December 5 (Submit through MyCourse)

1. [90%] Randomly create a 0/1 Knapsack problem as follows:

- Create  $n$  items where  $n$  is an integer randomly selected between 4 and 8. Display the selected  $n$  value.
- Create a list of  $n$  items where each item has profit  $p_i$  and weight  $w_i$  where 1)  $p_i$  is a positive integer randomly selected between 10 and 30; and 2)  $w_i$  is a positive integer randomly selected between 5 and 20. Set the capacity of the knapsack  $W = \text{floor}(0.6 * \sum_{i=1}^n w_i)$ .

To solve the created 0/1 knapsack problem, implement the following algorithms discussed in class:

- 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 together with their profits and weights.
- 2) [40%] Implement the refinement algorithm of dynamic programming approach (slides 38-40 of Ch12 lecture notes): (a) print the entries that are calculated in each row; (b) print the total profit and weight for the final solution; and (c) print the selected items together with their profits and weights for the final solution. Hint: you may need to define an extra variable to keep track of which items to be included in the knapsack comparing to the pseudo code of the general dynamic programming approach for the 0/1 knapsack problem (slide 33 of Ch12 lecture notes). If you have implemented the refinement algorithm of dynamic programming approach correctly, the total profit achieved by this algorithm must be equal to that achieved by the brute force method implemented in 1).
  - Note: You are supposed to implement the algorithm using the refinement algorithm of dynamic programming approach for the 0/1 knapsack problem. You will get no credit if you implement the general dynamic programming by calculating all the entries of matrix  $B$  (as shown in slide 33 of Ch12 lecture notes). We did not talk much about space complexity for this problem, and so it is acceptable if you define a full matrix  $B$  but set the entries that do not need to be computed to zero.
- 3) [40%] Implement the backtracking algorithm: (a) print the profit, weight and bound for each node according to the order that is first visited in the implicit pruned state space tree. (b) print the total profit and weight for the final solution; and (c) print the selected items together with their profits and weights for the final solution. Compare the result to the result you have got in 1). If you have implemented the backtracking algorithm correctly, the total profit achieved by this backtracking method must be equal to that achieved by the brute force method implemented in 1).

- Note: You are supposed to implement these algorithms correctly for **any 0/1 knapsack problems** as described above. If your program produces correct results for some 0/1 knapsack but doesn't for other knapsacks, you will get no credit.

2. [10%] Coding style: Write meaningful comments, while making your code structured, easy to read, and robust.

All programming must be done using **C or C++ in Linux** where your code will be tested. Create a tar file that includes (1) source code files, (2) a Makefile to produce an executable, and (3) a readme file that clearly describes how to run your code. Submit only the tar file through the Blackboard. The name of the tar file should be yourlastname\_yourfirstname\_proj5.tar (Do not use special characters like #, @, or &, because they have caused Blackboard problems in the past.) Suppose that your assignment files are under the directory of /your\_userid/yourlastname\_yourfirstname\_proj5/ and you are under that directory right now. To create a tar file under /your\_userid directory, do the following in Linux command line:

```
>cd ..
>tar cvf yourlastname_yourfirstname_proj5.tar yourlastname_yourfirstname_proj5
```

To view the content of the created tar file, do the following in Linux command line:

```
>tar tvf yourlastname_yourfirstname_proj5.tar
```

Finally, read the following policies carefully:

- All work must represent each individual student's own effort. 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 tool.
- 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.
- The instructor and TA will not read or debug your code. The instructor and TA will not take a look at an emailed code. If you need general directions, show your code to a TA during his office hours. The TA will not do programming or debugging for you though. The TA will only help you understand algorithms to be implemented and answer basic questions related to implementation.