Algorithms EECS 336 **Problem Set #2**
Date: April 16, 2018 **Due:** April 22, 2018

Submit your solution on Canvas.
Do not discuss these problems with other students. You should solve these problems on your own.

Problem 1. In this problem, we ask you to answer 8 questions. Please, submit your answers under the Quizzes section of Canvas.

I. Answer the following questions about the $O(\cdot)$ notation.

1. True or False: $n = O(e^n)$.
2. True or False: $\log n = O(n)$.
3. True or False: For all positive functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$, if $f(n) = O(g(n))$, then
$$(f(n))^2 = O((g(n))^2).$$
4. True or False: For all positive functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$, if $f(n) = O(g(n))$, then
$$e^{f(n)} = O(e^{g(n)}).$$

II. Find the Minimum Spanning Tree (MST) for each graph given on the last page of this homework and compute its cost.

5. Cost of the MST for Graph #1.
6. Cost of the MST for Graph #2.
7. Cost of the MST for Graph #3.
8. Cost of the MST for Graph #4.

Problem 2. In this problem, we ask you to design a greedy algorithm for handling overbooking. Suppose that an airline sold n tickets for a particular flight. However, there are only $k \leq n$ seats available on the plane. Consequently, the airline needs to select $(n - k)$ passengers who will take the next available flight. Suppose that if the passenger i is rebooked to another flight, the airline pays a penalty of p_i; if he or she takes the planned flight, the airline receives a revenue of r_i. The algorithm should find a set S of size $(n - k)$ that contains the IDs of passengers who should be rebooked to another flight so as to maximize the total revenue.

Formally, the problem is defined as follows. The algorithm receives a parameter k and two lists of numbers $r_1, \ldots, r_n$ and $p_1, \ldots, p_n$. The algorithm should find a set $S \subset \{1, \ldots, n\}$ of size $(n-k)$ so as to maximize the total revenue defined as follows:

$$\text{revenue}(S) = \sum_{i \notin S} r_i - \sum_{i \in S} p_i.$$

I. Design and describe a greedy algorithm for this problem.

II. Analyze its running time. To get a full credit for the problem, the running time of the algorithm must be $O(n \log n)$.

III. Prove that the algorithm is correct.

- The proof should use an exchange argument. Assume that the algorithm returns a set S_{ALG} and the optimal solution is S_{OPT}. Prove that if revenue(S_{OPT}) $\neq$ revenue(S_{ALG}), then there exist elements $a \in S_{ALG} \setminus S_{OPT}$ and $o \in S_{OPT}$ such that

$$\text{revenue}(S_{OPT} \cup \{a\} \setminus \{o\}) > \text{revenue}(S_{OPT}).$$

  Here, $S_{OPT} \cup \{a\} \setminus \{o\}$ is the set obtained from S_{OPT} by adding element a and removing element o.

- Use the inequality above to get a contradiction. Conclude that revenue(S_{OPT}) = revenue(S_{ALG}).

Hint: First, consider two special cases: (1) when all $r_i = 0$ and (2) when all $o_i = 0$.

Problem 3. In this problem, your goal is to implement a bracket matching algorithm for a new Wildcat IDE – an integrated development environment for the Wildcat language. This programming language has three types of brackets: (1) "(" and ")", (2) "{" and "}", (3) "[" and "]". We ask you to write the following function.

- `int MatchBracket (const std::string& str, int position)`

Given a string `str` and the position of a bracket `position`, this function should return the position of the matching bracket. The index of the first character in the string is 0. If the character `str[position]` is not a bracket or if there is no matching bracket for `str[position]` in `str`, then the function should return -1.

Example. *For string str="*`if (list[k].university() == 'NU') {print k}`*" and position* $= 3$*, the function should return 32. For position* $= 8$*, the function should return 10. For position* $= 23$*, the function should return 22. However, for string str="*`if (list[k).university() == 'NU') {print k}`*" and position* $= 8$*, the function should return -1;*

Instructions for the programming assignment. Download files:

- `student_code_2.h` – this file should contain your solution.
- `problem_solver_2.cpp` – this is the main file in the project (don't edit this file!).
- `test_framework.h` – this is a library responsible for reading and writing data files (don't edit this file!)
- `problem_set_2.dt` – this file contains test problems for your algorithm (don't edit this file!)

Place all files in a new folder/directory. Write your code in the function `MatchBracket`. Also, write your name in the function `GetStudentName`. Both functions are located in file `student_code_2.h`. Compile and run your code. To compile your code do the following.

- If you use GNU C++ compiler, type
  `g++ -std=c++11 problem_solver_2.cpp -o problem_solver_2`
- If you use CLang compiler, type
  `clang++ -std=c++11 problem_solver_2.cpp -o problem_solver_2`
- If you use Microsoft Visual C++ compiler, start `Developer Command Prompt` and type
  `cl /EHsc problem_solver_2.cpp`

Your compiler should be compatible with `C++11`. If you work in TLab, you need to start developer tools first: Type

- `scl enable devtoolset-4 bash`

Once you compile your code, start your program. Type `./problem_solver_2` on Unix or Mac and `problem_solver_2.exe` on Windows. Make sure that the executable is located in the same folder as file `problem_set_2.dt`. Your program will generate `solution_2.dat` that contains solutions to the problems from file `problem_set_2.dt`. If your code works correctly, you will get the following message:

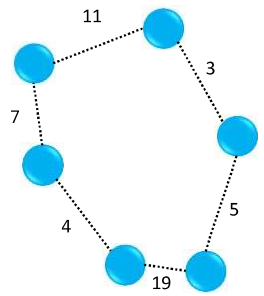
- Problem set 2. Your algorithm solved all test problems correctly. Congratulations!
- Don't forget to submit your source code and file `solution_2.dat` via Canvas.

If your code makes a mistake, you may get a message like this:

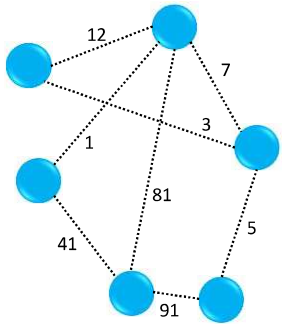
- `Problem set 2. Mistake in problem #15. Correct answer: 4. Your answer: 12.`

Finally, when your code is ready, submit files `student_code_2.h` and `solution_2.dat` via Canvas. Make sure that you are submitting the latest versions.

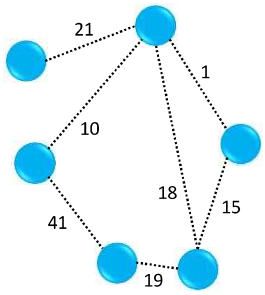
Remark: If you want to debug your code, please, type `./problem_solver_2 15` on Unix or Mac and `problem_solver_2.exe 15` on Windows. This command will call your function only on one problem – the problem #15 and thus let you debug your code on the problem where your program erred. Note that this command will not generate or update `solution_2.dat`. So before submitting your solution, you need to run your program without any command line arguments.



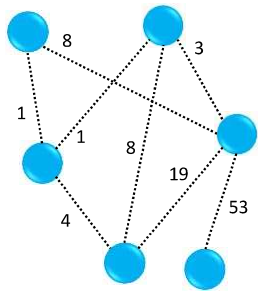
Graph #1



Graph #2



Graph #3



Graph #4