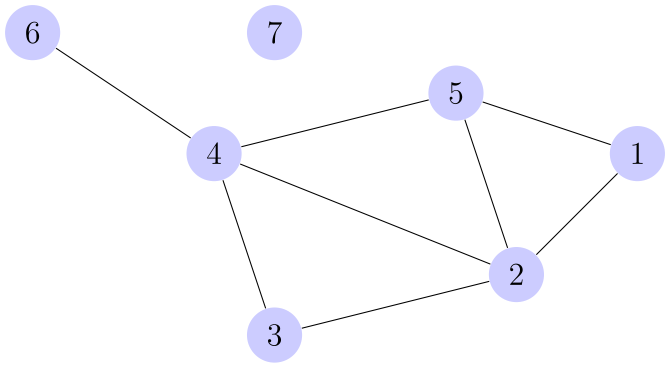
Fun with Algorithms

150 points

1. **Greedy Algorithms (40 pts)** - An *undirected* graph is usually defined by $G = (V, E)$ where V is a set of vertices and E is a set of edges. However, instead of saving the vertices and edges to a file, a student accidentally saved the number of edges each vertex has. In other words, for each $v \in V$, the number of edges was printed. Your goal is to verify whether or not the file can be a valid graph. You are to read in a file called **input1.txt** that contains n numbers: $e_1, e_2, e_3, \ldots, e_n$. You need to verify whether there exists a graph G such that v_1 has e_1 edges. v_2 has e_2 edges and so on. This must be done in polynomial time. There are some requirements G must follow. G must not have any self-looping edges. In other words, an edge (v, w) must have $v \neq w$. G must not have more than 1 edge between the same pair of vertices. The file will be formatted with each "number of edges" separated by a space. A vertex with no edges will have a number 0. The following is a sample file:
   2 4 2 4 3 1 0

   The above file can be represented with the following graph so if you read in the above file, you would output *yes*:

   However, the following file is not possible:
   2 4 2 4 3 2 0

   You do not need to create the actual graph. You simply need to output *yes* if a graph is possible, *no* otherwise. Your code must finish running in 60 seconds on a reasonably sized input (5000 vertices) or it will be considered incorrect.

2. **Greedy Algorithms (40 pts)** - You own a downtown conference room that overlooks Lake Michigan. Each person can make a request to reserve the conference room for 1 day at a time. The person gives the request in the following way: some unique request id, the amount willing to pay for the conference room for the day, the latest day in which the request must be fulfilled. Your goal is to maximize the amount of money you make. For example, consider the following list of requests:

1 100 4
2 50 3
3 75 2
4 25 1

In the above list of requests, all requests can be fulfilled. Request 4 can be fulfilled on day 1, request 3 on day 2, request 2 on day 3 and request 1 on day 4. However, consider the following requests:
1 100 4
2 50 2
3 75 2
4 25 1

In this case, the optimal set of requests to fulfil is requests 1, 2 and 3. Request 2 could be fulfilled on day 1, request 3 on day 2 and finally, request 1 on day 4. The 4th request is not fulfilled.

Your goal is to read in a list of requests and output a set of requests that were able to be fulfilled and the total amount earned by fulfilling those requests. The input will come from a file called **input2.txt**. All requests will be on their own line and will be in the following format: request id, amount willing to pay for one day in the conference room, latest day in which the request must be fulfilled. Your code must implement some kind of greedy algorithm and run as quickly as possible.

3. **Linear Programming (50 pts)** - You work for a company that manufactures clothing. You want to maximize the profit you can make per day given many constraints on materials and labor. For every pair of pants you create, you make a profit of \$10. For every shirt you create, you make a profit of \$7. In order to create a pair of pants, you need 4 buttons, 2 units of cloth and 3 units of denim. In order to create a shirt, you need 8 buttons, 1 unit of cloth and 2 units of denim. Your company can obtain 1000 units of cloth and 500 units of denim per day. You want to be sure your employees are doing about the same amount of work, therefore, you cannot make >= 20 more shirts than pants and you cannot make >= 20 more pants than shirts. You must give the linear program to solve this problem. What is the objective function? What are the constraints? You do not need to actually solve the problem with a linear programming solver.

4. **NP-Hardness (20 pts)** - Convert the following SAT formula into an equivalent 3SAT formula:

$$(x_1 \vee x_3 \vee x_5 \vee \overline{x_6} \vee x_7) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3} \vee x_4 \vee x_6 \vee \overline{x_7})$$