Stats 545: Homework 2

Due before midnight on Sunday, Sept 25.
All plots should have labelled axes and titles.

Important:`Rcode`, tables and figures should be part of a single .pdf or .html files from `R Markdown` and `knitr`. See the class reading lists for a short tutorial.Any derivations can also be in Markdown, in Latex or neatly written on paper which you can give to me.

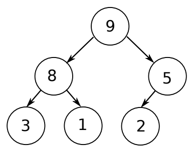
You are free to discuss homeworks with other students, but should write your own code and reports. If you collaborated significantly with anyone else, mention their names and the nature of the collaboration

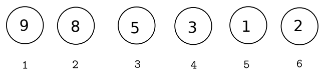
1 Problem 1: Implementing priority queues and sorting. [60 pts]

We will create two different implementations of a priority queue each with maximum capacity `LMAX`. While there's no need to limit the size of the queue, we will do this for simplicity, so that we can pre-allocate memory (increasing the length of a vector in `R` by 1 is an $O(n)$ operation: `http://musicallyut.blogspot.com/2012/07/pre-allocate-your-vectors.html`)

First we will implement an ordered doubly linked list. This will be an `R` list (`http://rforpublichealth.blogspot.com/2015/03/basics-of-lists.html`) consisting of 3 vectors: `data, next` and `prev` (all of length `LMAX`, containing the data, and indices to next and previous elements respectively). It will also have three scalars, `start, end` and `length`, pointing to the first and last elements, and giving the length of the linked-list.

1. Initially, your list will be empty, so that `start` = `end` = `length` = 0. Write a function `make_dll` that returns an empty doubly linked-list. [2pts]
2. Write a function `get_dll_max` that accepts a linked list as input and returns its maximum value. [3pts]
3. Write a function `remove_dll_max` to remove the maximum value from the input linked-list. Remember to adjust `prev, next, start` and `end`. [5pts]
4. Write a function `insert_dll_element` to insert an element into your list. Recall that since your linked list is sorted, you first need to find the correct location to insert, and then actually insert the element. If your list already has `LMAX` elements, return an error. [10pts]





Next, we will organize a heap as a vector, organizing its elements sequentially as in the right subfigure above.

5. What is the index of the parent node of element i of the vector? What are the indices of the left and right child nodes of element i of the vector? [5pts]

We will implement a heap as a list consisting of a vector of length `LMAX` and a scalar `length` giving the actual number of elements in the queue.

6. Write a function `make_heap` to return an empty heap. [2pts]
7. Write a function to return the maximum element of the heap. [3pts]
8. Write a function to remove the maximum element of the heap. [10pts]
9. Write a function to insert a new element into the heap. For the last two, you can create additional helper functions if you want to. [10pts]
10. Finally, write two functions to sort a vector of number of numbers using each of the two priority queues (equivalently, you can write a single function that takes as an argument which priority queue to use). The functions should first sequentially add elements to the queue, and then sequentially remove the maximum element. Demonstrate that these work on an example with 20 random numbers. [10pts]

2 Problem 2: The knapsack problem [20pts]

Here we will implement the unbounded knapsack problem (unbounded refers to the fact we have an infinite number of objects of each type).

1. Write a function that accepts as input two vectors of the same length, `w` and `v`, the ith elements of which give the weight and value of the ith object. The function also accepts a scalar `W_knapsack` giving the capacity of the knapsack. The function should return two objects, a scalar `V_knapsack`, giving the maximum value of the knapsack, and a vector `obj_count`, the ith element of which gives the number of objects of type i contained in the bag when it is filled optimally. [15pts]
2. Test your function for the case `w = (1,2,3,4,5,6), v = (1,8,10,10,19,25)` and `W = 25`. [5pts]

3 Problem 3: Markov chains [20pts]

Consider an N-state Markov chain. Call the state at time t as S_t, which can take values from 1 to N (i.e. $S_t \in \{1, \ldots, N\}$). The Markov chain has two parameters, an initial distribution over states π^1, and a transition matrix A. The former is an $N \times 1$ probability vector with $\pi^1_i = P(S_1 = i)$. The latter is an $N \times N$ stochastic matrix (i.e. a matrix whose rows are nonnegative and sum to 1). A_{ij}, the $(i,j)^{th}$ element of this matrix gives the probability of moving from state i to state j. Thus $P(S_{t+1} = j|S_t = i) = A_{ij}$.

Write down as functions of A and π^1:

1. $P(S_2 = j, S_1 = i)$ [2pts]
2. $P(S_2 = j)$ [3pts]

Let π^t be an $N \times 1$ column-vector giving the probability over states at time t.

3. Write π^2 as a function of of π^1 and A. Use matrix notation (i.e no summations). [3pts]
4. How many summations and multiplications does this involve? This gives the cost is big-O notation, write it down (i.e. is it $O(N), O(N^2), O(N^3), O(2^N)$ or what?). [3pts]
5. Write π^t as a function of of π^1 and A. Use matrix notation (write A^2 for the multiplication of A with itself, i.e. $A^2 = AA$. This notation is slightly ugly, since A^t means multiplying A with itself t times, while π^t means the probability vector at time t. This is ok, since it doesn't make sense to multiply a vector with itself). Write down the cost of calculating this quantity in big-O notation (now it is a function of N and t). [3pts]

Consider a complete sequence of states $\mathbf{S} = (S_1, S_2, \ldots, S_{T-1}, S_T)$.

6. How many such sequences are there? [3pts]

7. Imagine we had a big table giving the probability of each sequence that is, someone has already calculated it for us. If for some time t we wanted to calculate the probability $P(S_t = i)$ by brute force, we could marginalize out the other states:

$$P(S_t = i) = \sum_{\mathbf{S} \text{ such that } S_t = i} P(\mathbf{S}).$$

   How many summations does this approach require? If we wanted to calculate the entire vector $P(S_t)$, how many summations does that require. [3pts]

Comment: the moral here is that it is expensive to store and calculate marginal probabilities for a general T-component sequence. By exploiting the Markov structure, we can save time and space