ASSIGNMENT 3:
CSE 4131: ALGORITHM DESIGN – II

You are allowed to use only those concepts which are covered in the lecture classes, till date.
Plagiarized assignments will be given zero mark.

Submission deadline: Open time

Submit the hard copy of the assignment.

SECTION – A (Local Search)

Q1. (a) Using gradient-descent method, how to get the optimal solution of vertex-cover problem to an instance with an n-node graph with no edges? Find out both global and local optimal solution.

(b) Using gradient-descent method, how to get the optimal solution of vertex-cover problem to an instance with a "star graph" G, consisting of nodes x_1, y_1, y_2 , . . . , y_{n-1} , with an edge from x_1 to each y_i? Find out both global and local optimal solution.

(c) Using gradient-descent method, how to get the optimal solution of vertex-cover problem to an instance with a bipartite graph G with nodes x_1 , x_2 , . . . , x_k and y_1 , y_2 , . . . , y_l , where $k < l$, and there is an edge from every node of the form x_i to every node of the form y_j? Find out both global and local optimal solution.

Q2. Consider a statement – "In a general energy landscape, there may be a very large number of local minima that make it hard to find the global minimum." Justify the statement with proper reasoning.

Q3. Write and explain Hopfield's state-flipping algorithm.

Q4. "Simulated Annealing is a heuristic search procedure that allows occasional transitions leadind to more expensive (and hence inferior) solutions, but it helps keep our search from getting stuck in local optima." State TRUE/FALSE with proper reasoning.

Q5. Given an undirected interger-weighted graph $G=(V_s,E_w)$ that represents a Hopfield Neural Network where set V_s represents the set of all vertices with their states as pair of (vertex,state) and E_w represents the set of all edges with their edge-weights as pair of (edge,edge-weight) which is as follows:

$V_s = \{(v_1,-1),(v_2,+1),(v_3,+1),(v_4,+1),(v_5,-1)\}$ and

$E_w = \{((v_1,v_2),-10),((v_1,v_3),8),((v_2,v_3),-4),((v_4,v_3),-1),((v_5,v_3),-1)\}$

(a) For the above given configuration of the Hopfield neural network, find the sum of weights of all good edges.

(b) Using the State-flipping algorithm, show all the steps to find a stable configuration from the given configuration of the Hopfield neural network.

(c) Give a suitable bound for the number of iterations for the State-flipping algorithm to find a stable configuration.

Q6. "Each edge of the Hopfield neural network imposes a requirement on its endpoints: If u is joined to v by an edge of negative weight, then u and v want to have the same state, while if u is joined to v by an edge of positive weight, then u and v want to have opposite states." Provide a counter-example to this statement.

Q7. Given a directed Hopfield neural network with nodes a, b, c, with directed edges (a, b), (b, c), (c, a), all of weight 1, find a stable configuration from any random intial configuration. Mention the challenges in the process.

SECTION – B (Randomized Algorithm)

Q1. Give a critical comparison between the two distinct ways about the implementation of the random processes in the context of computation.

Q2. The keys 13, 19, 14, 3,4, 24, 6 and 16 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing. Find the resultant hash table?

Q3. Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions?

Q4. Universal hash function:

A universal family of hash functionsis a set of hash functions H mapping a universe U to the set {0,1, .., n– 1} such that:

- For any pair of elements $u \neq v$: $Pr_{h \in H}[h(u)=h(v)] \leq 1/n$
- Can select random h efficiently and can compute h(u) efficiently.

Ex. Given U = { a, b, c, d, e, f }, n = 2, find whether H is a universal family of hash functions or not in the following cases:

(a) $H = \{h_1,h_2\}$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

(b) $H = \{h_1,h_2,h_3,h_4\}$

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1
$h_3(x)$	0	0	1	0	1	1
$h_4(x)$	1	0	0	1	1	0

Q5. Let A={0,1} be an alphabet from which a given string S is taken. Let char(c) be a function that maps each symbol of the alphabet to a unique integer from 0 to |A|−1. The function $H(S)=\left(\sum_{i=0}^{|S|-1}|A|^{|S|-(i+1)}char(s(i))\right) mod\, m$ maps each string to a unique integer by treating the characters of the string as digits in a base−|A| number system. For the strings: S_1 = "1001", S_2 = "0110", S_3 = "1100", S_4 = "1010", S_5 = "0101" and size of the hash table,i.e m = 7.

(a) Design a hash table with open addressing(with sequential probing).

(b) Design a hash table with chaining.

(c)(i) Find the number of collisions.

(ii)Find the average number of string comparisons for successful search.

Q6. Select(S, k)

```
{       choose a splitter $a_i \in S$, uniformly at random
        for each element $a_j$ of S {
                if $a_j < a_i$ then put aj in $S^-$
                if $a_j > a_i$ then put aj in $S^+$
        }
        if $|S^-| = k - 1$ then
                return $a_i$      // the splitter ai was in fact the desired answer
        else if $|S^-| \geq k$ then
                Select($S^-$,k)  // the kth largest element lies in S−
        else // suppose $|S^-| = l < k - 1$
                Select($S^+$,k)    // the kth largest element lies in $S^+$
}
```

(a) "Regardless of how the splitter is chosen, the algorithm Select(S,k) above returns the kth largest element of S." − TRUE/FALSE. Justify your answer.

(b) If we could always choose the median as the splitter, then we could show a linear bound on the running time. Let cn be the running time for Select(), not counting the time for the recursive call. Then, with medians as splitters, give an upper bound on the running time T(n).

(c) Choosing a "well−centered" splitter, if we had a way to choose splitters a_i uniformly at random such that there were at least εn elements both larger and smaller than a_i , for any fixed constant $\varepsilon > 0$, then give an upper bound on the running time T(n).

Q7. In a hashed implementation of a dictionary data structure, the time required for Lookup(u) is proportional to the time to compute the hash address h(u) plus the length of the linked list at H[h(u)], where H[] is the hash table. Explain this implementation scenario with a suitable example.

Q8. Mention the principal criteria to choose a good hash function. How this selection implements randomization in the dictionary data structure?