CSE 4131: ALGORITHM DESIGN 2

Assignment 3: SOLUTION HINTS

Text book: Algorithm Design by Eva Tardos, J. Kleinberg, Pearson Publication

Instructions: Read all the questions carefully multiple times and try to understand the meaning of the questions. Then start to solve by the help of these hints. Still if you have any doubt regarding questions then feel free to discuss with your concern faculty.

Sl. No.	Questions

1.	Greedy-Balance (n, m, t[1...n]) {
	    Start with no jobs assigned
	    Set Ti = 0 and $A_i = \emptyset$ for all machines M_i
	    For j = 1, . . . , n do {
	        Let M_i be a machine that achieves the minimum $\min_k T_k$
	        Assign job j to machine M_i
	        Set $A_i \leftarrow A_i \cup \{j\}$
	        Set $T_i \leftarrow T_i + t_j$
	    }
	    return $A[1], A[2], \dots, A[m]$
	}
	a. Show that algorithm Greedy-Balance produces an assignment of jobs to machines with makespan $T \leq 2T^*$.
	b. What will be the resulting makespan of running this greedy algorithm on a sequence of six jobs with processing times 12, 13, 14, 16, 12, 12 for m=3 identical machines?

Hints. 1a	See **lemma 11.3** in the text book. Page no. 602.
Hints. 1b	Resulting makespan= 28. [Do it on your own and compute each steps.]
2.	Consider an example scenario of Center-Selection problem with only two sites s and z, and k = 2. Assume that s and z are located in the plane, with distance equal to the standard Euclidean distance in the plane, and that any point in the plane is an option for placing a center. Let d be the distance between s and z. Then find the best location for (i) Case1: a single center c_1 and (ii) Case2: then subsequent best location for second center c_2. In each case, also find the optimal covering radius r©.
Hints.	See the text book in **page no. 607.**

3.	Given an approximation algorithm for weighted-set-cover problem as below:
	    Greedy-Set-Cover() {
	        Start with R = U and no sets selected
	        while ($R \neq \Phi$)

```
                Select set Si that minimizes $w_i/|S_i \cap R|$
                    $S = S \cup \{S_i\}$
                Delete set $S_i$ from R.    // $R = R - \{S_i\}$
            }
            Return the selected sets S
    }
```

Using the above algorithm, find the solution S for the following set-cover problem instance and compare this solution with the optimal solution S*. Set-Cover instance with U={1,2,3,4,5,6,7,8}, and list of subsets such as S1={1,3,5,7}, S2={2,4,6,8}, S3={1}, S4={2}, S5={3,4}, S6={5,6,7,8} and weight array w[] = {1+ε, 1+ε, 1,1,1,1}, where ε is a very small value between 0 and 1.

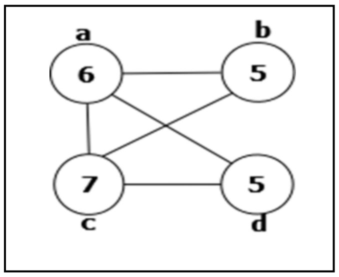
Hints.

- The solution by the given algorithm is, S={S6, S5, S3, S4}. Here W(S)=4
- The optimal solution S*={S1, S2}. Here W(S)= 2+2 ε < 4 , Since ε is a very small value between 0 and 1.

Compute each steps of the problem using given algorithm and check your solution.

4. Given an approximation algorithm for weighted-vertex-cover problem as below:

```
Vertex-Cover-Approx( G, w ) {
        Set $p_e = 0$ for all $e \in E$
        While (there is an edge $e = (i,j)$ such that neither i nor j is tight) {
                Select such an edge $e$
                Increase $p_e$ without violating fairness
        }
        Let S be the set of all tight nodes
        Return S
}
```

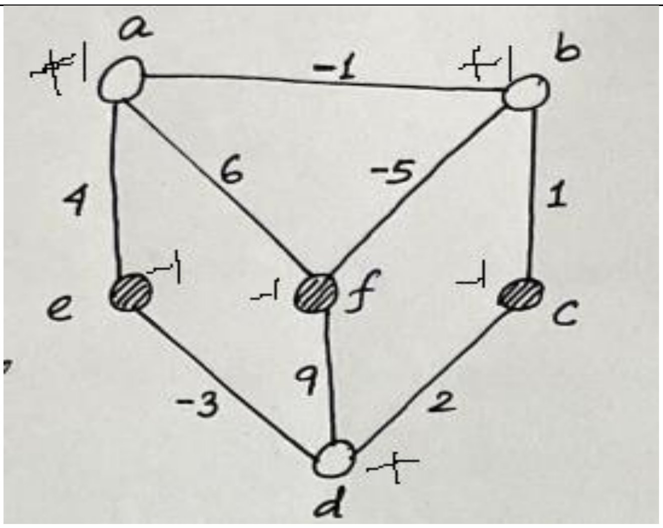
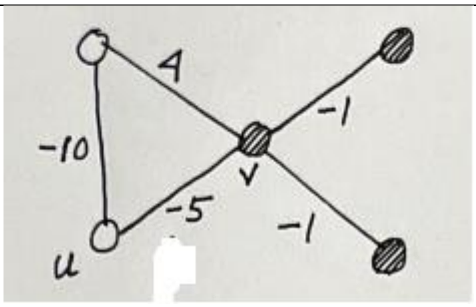


a. Using the above algorithm based on pricing method, find the weighted-vertex-cover for the given graph with four vertices having vertex weights/costs 6, 5, 7 and 5. Compare your solution with the optimal solution for this example.
b. Prove that: "For any vertex cover S, and any nonnegative and fair prices p_e , we have $\sum_{e \in E} p_e \leq W(S)$ " (Fairness lemma).
c. Show that "the set S returned by the above algorithm is a vertex cover, and its cost is at most twice the minimum cost of any vertex cover."

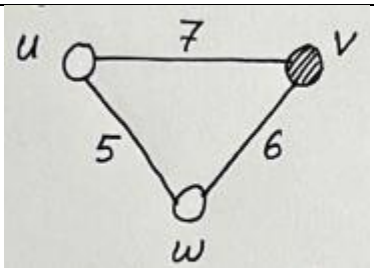
Hints 4a

- The solution by the given algorithm is, S={a, c, d}. Here W(S)=18 [This answer may very depends on selection of arbitrary edges.]
- The optimal solution S*={a, c}. Here W(S)= 13

Compute each steps of the problem using given algorithm and check your solution.

Hints 4b	See **lemma 11.13** in the text book. **Page no. 620-621.**
Hints 4c	See **lemma 11.15** in the text book. **Page no. 623.**
5.	Find whether the given configuration S is stable or not. If not, then make it stable by using the state-flipping algorithm. An undirected weighted graph defines the configuration S consists of a set of nodes *V={a, b, c, d, e, f}*, set of edges *E={(a, b), (a, e), (a, f), (b, f), (b, c), (c, d), (e, d), (f, d)}*, associated edge weights *W={-1, 4, 6, -5, 1, 2, -3, 9}*, and the node assignment $S_e = S_f = S_c = -1$ for all other nodes it is +1.
Hints	 a +1 -1 +1 b 6 -5 4 1 -1 -1 -1 e f c -3 9 2 + d First draw the configuration. Then identify characteristics of each nodes and edges. After that apply your state flipping algorithm on this graph. Follow the state flipping algorithm mentioned in your text book page number-761.
6.	Can there be bad edges in a stable configuration S of a Hopfield neural network? Justify with an example. An edge $e = (u, v)$ is a bad edge, if $w_e S_u S_v > 0$, where w_e is the weight of edge e, and $S_u = \pm 1$ is the node assignment for a node u.
Hints	 Now check the given conditions.
7.	In a Hopfield neural network, for a given configuration S, An edge $e = (u, v)$ having weight $w_e < 0$ then node u and v are in the same state represented by s_u=*1* and s_v=*1* (or s_u=*-1* and s_v=*-1)*. However, in general, a configuration may not respect the constraints. Justify with an example.

Hints



Now check the given conditions.

8. Apply gradient descent local search algorithm to find the vertex cover from the given complete bipartite graph G=(V, E), where V=L∪R, L={a, b, c, d, e}, and R={m, n, o, p, q, r}. Here complete bipartite graph means every vertices of L connected to the each and every vertices of R. Can gradient descent always guarantee finding the global optimum from a solution space? If not, why?

Hints Do it on your own. See the text book in **page no. 753.**

9. Apply gradient descent local search algorithm to find the vertex cover from the given graph G=(V, E), where V={a, b, c, d} and *E={(a,b),(c,d),(a,c),(b,d),(b,c)}*. Find all possible local solutions. Can gradient descent always guarantee finding the global optimum from a solution space? If not, why?

Hints Do it on your own. Follow the gradient descent local search algorithm.

11\.

```
Select(S, k)
{Choose a splitter $a_i \in S$.
For each element $a_j$ of S {
      if aj < ai then put aj in S−
      if aj > ai then put aj in S+
}
if |S−| = k – 1 then
     return ai                 // the splitter ai was in fact the desired answer
else if |S−| ≥ k then
        Select(S-,k)        // the kth largest element lies in S−
      else                     // suppose |S−| = t < 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).

Hints Go through the text book. See **lemma 13.17** in the text book. **Page no. 683.**