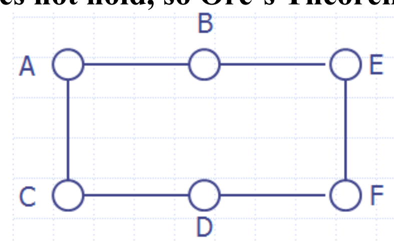
Lab 14

1. Ore's Theorem implies that graphs with "many edges" tend to be Hamiltonian. Is it true that every dense graph is Hamiltonian? Prove your answer.

   Solution: No. We have already showed that it is possible for a dense graph to be disconnected, but every Hamiltonian graph is connected.

2. Answer the following questions about the graph G having n = 6 vertices, below.
   a. Is G Hamiltonian? **Yes**
   b. Can you find two non-adjacent vertices the sum of whose degrees is less than 6? **Yes**
   c. Do these facts contradict Ore's Theorem? Explain. **No, Ore's Theorem is concerned only with consequences of the condition that the sum of degrees of nonadjacent vertices is at least the size of n; in this example, this condition does not hold, so Ore's Theorem does not apply.**



3. Show that TSP is NP-complete. (Hint: use the relationship between TSP and HamiltonianCycle discussed in the slides. You may assume that the HamiltonianCycle problem is NP-complete.)

   Solution:
   First show TSP is in NP.
   Suppose R is an NP problem. We must show that $R \xrightarrow{\text{poly}} TSP$. Notice that

   $$R \xrightarrow{\text{poly}} HC \xrightarrow{\text{poly}} TSP$$

   The first is because HC is NP-complete; the second is shown in the lecture.

   So $R \xrightarrow{\text{poly}} TSP$.

4. Recall from a previous lab the definition of the Knapsack problem. Show that the SubsetSum problem is polynomial reducible Knapsack. Assuming that you know SubsetSum is NP-complete (this is indeed true), explain the steps of logic that verify that Knapsack must also be NP-complete.

   Solution:

   Let SS be a SubsetSum instance of size n consisting of the set S = $\{s_0, s_1, \ldots, s_{n-1}\}$ of positive integers and a non-negative integer k. We show how to transform S into a Knapsack instance KN in polynomial time. We must provide a set of items with weights and values, and maximum weight W and minimum value V.

   We define KN as follows: Let the set S from SS be the KN set of items. For each i, let $w_i = s_i$ and let $v_i = s_i$. Let W = k and let V = k. These definitions specify an Knapsack instance of size O(n). We must show that a solution to SS yields a solution to KN, and conversely.

   Verification: Solution to SS ⇒ Solution to KN

   Suppose T is a solution to SS (T is a subset of S whose sum is k). We show T is also a solution to KN. We must show

$$\sum_{s_i \in T} w_i \leq W \text{ and } \sum_{s_i \in T} v_i \geq V.$$

   Since T is a solution to SS, we know

$$\sum_{s_i \in T} s_i = k.$$

   It follows that

$$\sum_{s_i \in T} w_i = \sum_{s_i \in T} s_i = k = W$$

   and

$$\sum_{s_i \in T} v_i = \sum_{s_i \in T} s_i = k = V$$

   as required. We have shown that a solution to SS yields a solution to KN.

   Verification: Solution to KN ⇒ Solution to SS

   Suppose T is a solution to KN (T is a subset of S the sum of whose weights is ≤ W and sum of whose values is ≥ V). We show T is also a solution to SS. We must show that

$$\sum_{s_i \in T} s_i = k.$$

   Since T is a solution to KN we have

$$\sum_{s_i \in T} s_i = \sum_{s_i \in T} w_i \leq W = k$$

and also

$$\sum_{s_i \in T} s_i = \sum_{s_i \in T} v_i \geq V = k$$

which, together, establish the desired result. We have shown that a solution to KN yields a solution to SS.

5. Show that the worst case for VertexCoverApprox can happen by giving an example of a graph G which has these properties:
   a. G has a smallest vertex cover of size *s*
   b. VertexCoverApprox outputs size 2*s as its approximation to optimal size.

   Solution:

   Consider the following disconnected graph with 2 edges and 4 vertices:

   A — B
   C — D

   The smallest vertex cover has size 2 – an example of such a vertex cover is {A, C}. However, the output of the VertexCoverApprox algorithm is {A, B, C, D}, a cover that has exactly twice ths size of a minimal cover.

6. Find an O(n) algorithm that does the following: Given a size n input array of integers, output the first numbers in the array (from left to right) whose sum is exactly 10 (or indicate that no such numbers can be found).

   Solution: Let S = {$s_0, s_1, . . ., s_{n-1}$} be the array elements in the original order. Fill in the memoization table for the SubsetSum problem given by S and k = 10. As the table is filled in from row 0 through row n – 1, look for a solution T to appear in any cell in the column headed by 10. Suppose the first such solution occurs in the row labeled by si. This means that T is a subset of $s_0 \ldots s_i$ whose sum is 10. Notice that no set of numbers lying in $s_0 \ldots s_{i-1}$ sums to 10 since the s_i row is the first in which such a solution occurs. Therefore, T is the solution to the problem. The running time to fill in the table and watch for the occurrence of T is O(10n)= O(n).