99 學年度第 2 學期資格考 演算法

1. (28% = 7*4%) Compare each pair of terms by giving full descriptions for them:
   (a) Upper bound vs. supremum
   (b) Complete graph vs. bipartite graph
   (c) 0/1 vs. continuous Knapsack problems
   (d) Hamiltonian cycle vs. Euler cycle
   (e) Simple path vs. simple cycle
   (f) Adjacency matrix vs. isomorphic graphs
   (g) Tree vs. heap
2. (20% = 2+2+2+6+2+2+4%) What is the worst-case time of *quicksort*? In what situation will it happen? What technique can be applied to avoid always selecting bad partition elements? Show that the expected run time of *quicksort* is $\Theta(n \lg n)$ with the above technique. What is a stable sort? Give a simple example to show that *quicksort* is not a stable sort. Show how the *quicksort* algorithm sorts the following array (in nondecreasing order)

   75, 6, 24, 27, 15, 8, 99, 72, 23, 8, 57, 62, 19
3. (20%) Compare the techniques of divide-and-conquer and dynamic programming in details. Give examples to demonstrate or support your arguments.
4. (10% = 6+4%) Suppose our character set is {A, B, C, D, E, F} and each character appears in the file the number of times indicated in Table 1.
   (a) Construct the Huffman coding tree.
   (b) Encode the strings “DEAF” and “BECAD”.
5. (22% = 14+8%) Trace *Floyd*’s algorithm to find the weight of the shortest path of each pair of vertices for the graph of Figure 1. Show all $A^{(i)}$ and the *next* matrices. Trace the *Finding-a-Shortest-Path* algorithm with the *next* matrix to give the path of each pair of vertices. The two algorithms are given in the next page.

Character	A	B	C	D	E	F
Frequency	16	5	12	17	10	25

Table 1

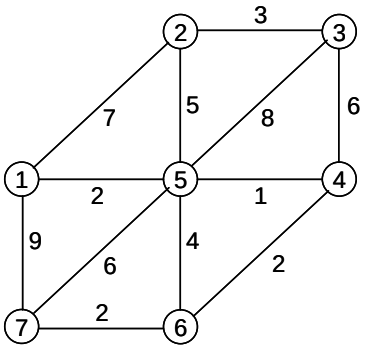

Figure 1