COMP20007 DESIGN OF ALGORITHMS

Week 4 Workshop

Tutorial

1. Subset-sum problem Design an exhaustive-search algorithm to solve the following problem:

Given a set S of n positive integers and a positive integer t, does there exist a subset $S' \subseteq S$ such that the sum of the elements in S' equals t, *i.e.*,

$$\sum_{i \in S'} i = t.$$

If so, output the elements of this subset S'.

Assume that the set is provided as an array of length n. An example input may be $S =$ `[1, 8, 4, 2, 9]` and $t = 7$, in which case the answer is YES and the subset would be $S' =$ `[1, 4, 2]`.

What is the time complexity of your algorithm?

2. Partition problem Design an exhaustive-search algorithm to solve the following problem:

Given a set S can we find a partition (*i.e.*, two disjoint subsets A, B such that all elements are in either A or B) such that the sum of the elements in each set are equal, *i.e.*,

$$\sum_{i \in A} i = \sum_{j \in B} j.$$

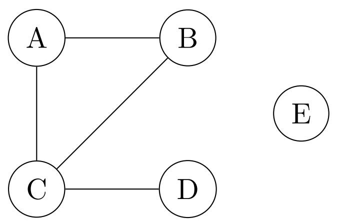
If so, which elements are in A and which are in B?

Again, assume S is given as an array. For example for $S =$ `[1, 8, 4, 2, 9]` one valid solution is $A =$ `[1, 2, 9]` and $B =$ `[8, 4]`.

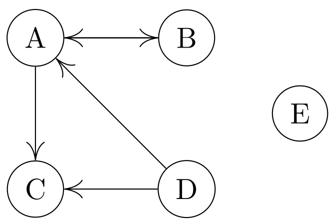
Can we make use of the algorithm from *Question 1.*? What's the time complexity of this algorithm?

3. Graph representations Consider the following graphs.

(a) An undirected graph:



(b) A directed graph:



Give their representations as:

(i) adjacency lists (ii) adjacency matrices (iii) sets of vertices and edges

Which node from (a) has the highest degree? Which node from (b) has the highest in-degree?

4. Graph representations continued Different graph representations are favourable for different applications. What is the time complexity of the following operations if we use (i) adjacency lists (ii) adjacency matrices or (iii) sets of vertices and edges?

(a) determining whether the graph is *complete*

(b) determining whether the graph has an *isolated node*

Assume that the graphs are undirected. A *complete* graph is one in which there is an edge between every pair of nodes. An *isolated node* is a node which is not adjacent to any other node.

Assume that the graph has n vertices and m edges.

5. Sparse and dense graphs (optional) We consider a graph to be *sparse* if the number of edges, m, is order $\Theta(n)$. On the other hand we say a graph is *dense* if $m \in \Theta(n^2)$.

Give examples of types of graphs which are sparse and types which are dense.

What is the space complexity (in terms of n) of a sparse graph if we store it using:

(i) adjacency lists (ii) adjacency matrix (iii) sets of vertices and edges

Computer Lab

In assignment 1 you will need to create and provide a Makefile, and you'll be required to test and submit your program using the School of Engineering `dimefox` servers.

1. Make and Makefiles If you haven't already, take a look back to last week's lab and ensure that you know how to create a `Makefile` and use `make`.

2. Dimefox Copy the code and Makefile from last weeks lab onto the `dimefox` server. Compile and run your code on the server manually, and then also using the Makefile you created.

The *Dimefox Instructions* document on the LMS will help with this task.

3. Linked List Module

This question is the same as the final question from last week. We'll spend some time this week making sure we can all write and create our own linked list module.

We'll use linked lists for many algorithms and data structures which are covered in this class, and it would be nice if we had some C code to use whenever we require a linked list in our programs.

In this question you should write a linked list module, comprised of `list.c` and `list.h`.

There will be some decisions you must make when you design your modules, for instance will your linked list be singly- or doubly-linked? Will you have a single node structure, or a linked list structure which contains more information about your list (*e.g.*, the head, tail and size of your list)?

Create functionality to insert and delete at the front and end of your list, find out how many elements are in the linked list and to free the linked list.

Once you've created your linked list module, create a main program to test your linked list.