MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

159.201 Algorithms & Data Structures
Summer 2014

Tutorial 7

1. For the following arrays:

array 1 = {1,2,3,4,5} (already sorted)

array 2 = {5,4,3,2,1} (reverse order)

array 3 = {3,4,2,5,1} (random order)

compile the number of comparisons used for each of the following algorithms:

	N=5		
	Random	reverse	sorted
Selection	10	10	10
Insertion	8	10	4
Bubble	20	20	4

Discuss the differences between the algorithms.

Answer: Firstly we should fit the functions with comparison counters.

```
long int comparison=0;

void selectionsort(vector<int> &varray,int n){
	int pass;int min;
	for (pass=0; pass<n-1; pass++) {
		min = pass;  // min is an index
		for (int i = pass + 1; i < n; i++) {
			comparison++;
			if (varray[i] < varray[min]) { // a comparison here
			  min = i;
			}
		}
		swap(&varray[min], &varray[pass]);
	}
}

void insertionsort(vector<int> &varray,int n){
	int temp;
	int i;
	for (int pass=0;pass<n-1; pass++) {
		temp=varray[pass+1];
		for (i=pass + 1; i > 0; i--)
		{
			comparison++;//compare array elements
			if(varray[i-1]>temp){
				varray[i] = varray[i-1];  // shuffling
			}
			else break;
		}
		varray[i] = temp;
	}
}

void bubblesort(vector<int> &varray,int n){
	int i;
	int pass;
```

```
        bool swapping = true;
        while (swapping) {
                swapping = false;
                for (i = 0; i < n-1; i++) {//don't look at the last one
                        comparison++;
                        if (varray[i] > varray[i+1]) {//comparison
                                swap(&varray[i], &varray[i + 1]);
                                swapping = true;
                        }
                }
                pass++;
        }
}
```

2. Write an algorithm (not a C++ program) that implements a sorting algorithm based on the Heap. There are two phases:

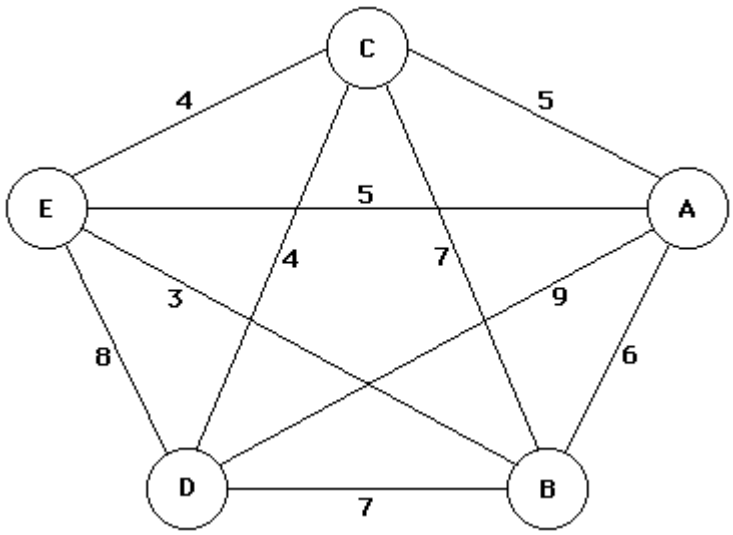
phase 1 reads the data in some order and builds the Heap (using Insert()).

phase 2 deletes the root of the Heap, and copy the deleted node to a temporary array.

Answer:

```
Algorithm HeapSort (Heap A, SourceArray data, NumberofElements n)
   Require: integer i
1  for(i=0 to n-1){
2    A.InsertHeap(data[i]);
3  }
4  for(i=n-1 to 0){
5    data[i]=A.DeleteRoot();
6  }
```

3. Use the graph below to solve the Travelling Salesperson Problem, starting at E. The total "distance travelled" (sum of weights) must be <= 26.



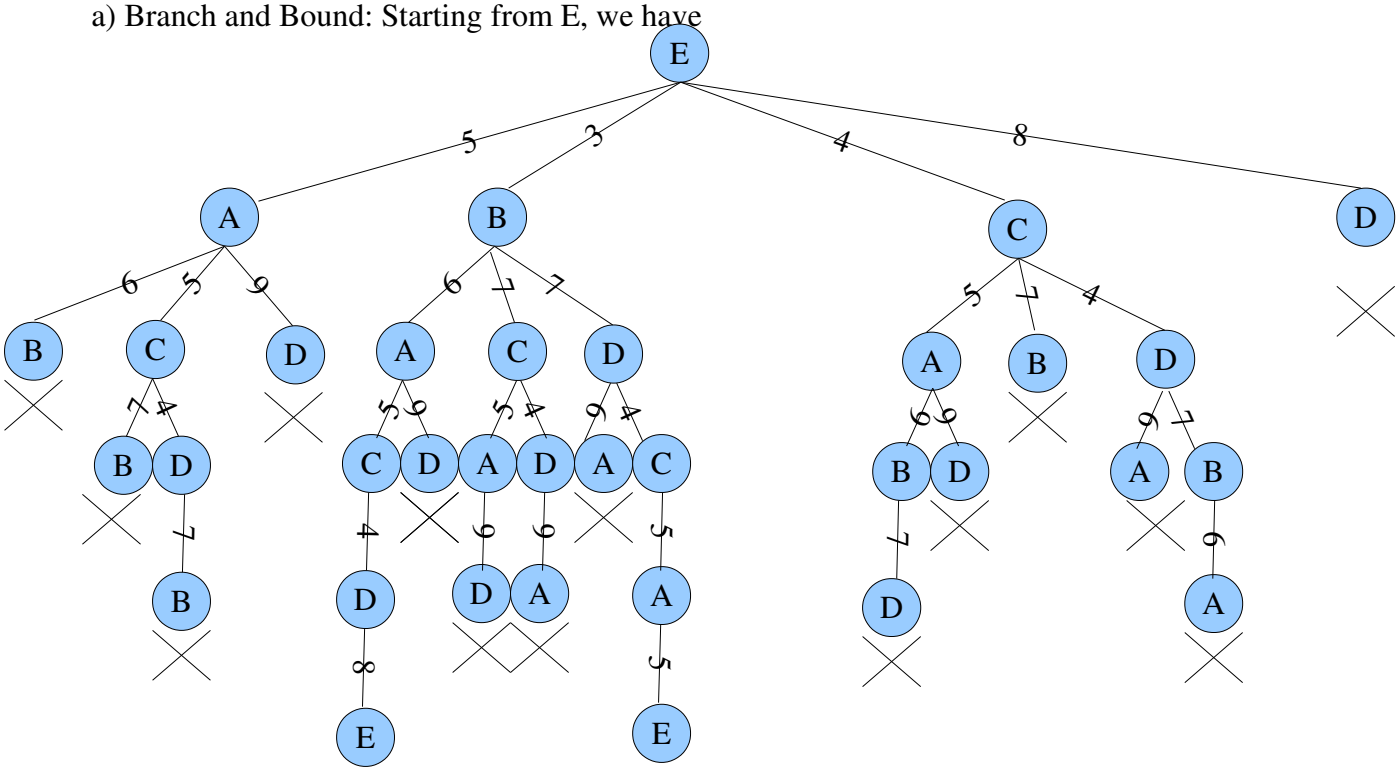
(a) Use the Branch & Bound method. Start with a bound of 5 and increase it by 5 at each step.

(b) Use a Greedy Algorithm which always takes the shortest available path.

Comment on the differences between (a) and (b).

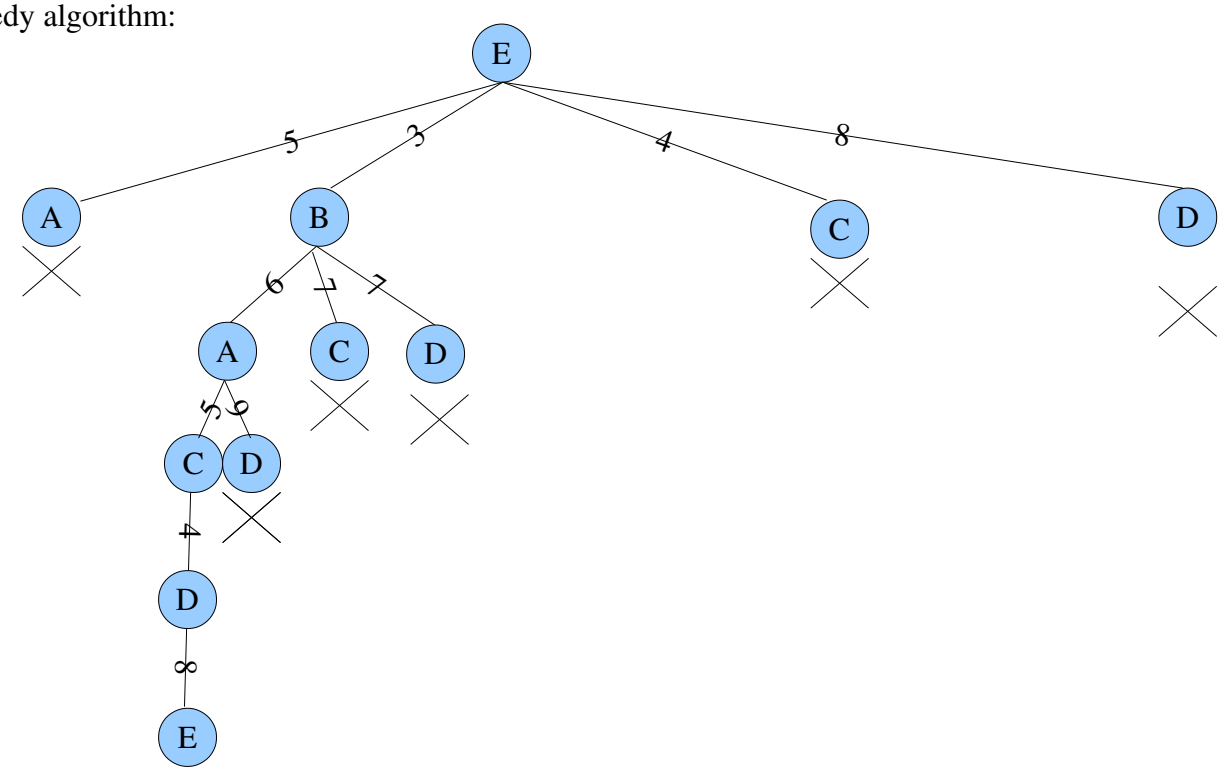
Answer:

a) Branch and Bound: Starting from E, we have



We have E-B-A-C-D-E with cost 26, or E-B-D-C-A-E with cost 24. The last path is shorter.

b) greedy algorithm:



We found E-B-A-C-D-E with cost 26. Although much faster, the greedy algorithm may not find the shortest path, nor some of the shorter paths such as the one found with Branch and Bound.