UWI

Design and Analysis of Algorithms

Tutorial 10 - Sample Solutions

Branch-and-Bound Technique

1. Solve the assignment problem using the branch-and-bound technique for the instance shown in table 1.

	Job 1	Job 2	Job 3
Person 1	5	7	9
Person 2	3	8	5
Person 3	7	4	9

Table 1: A small instance of the assignment problem

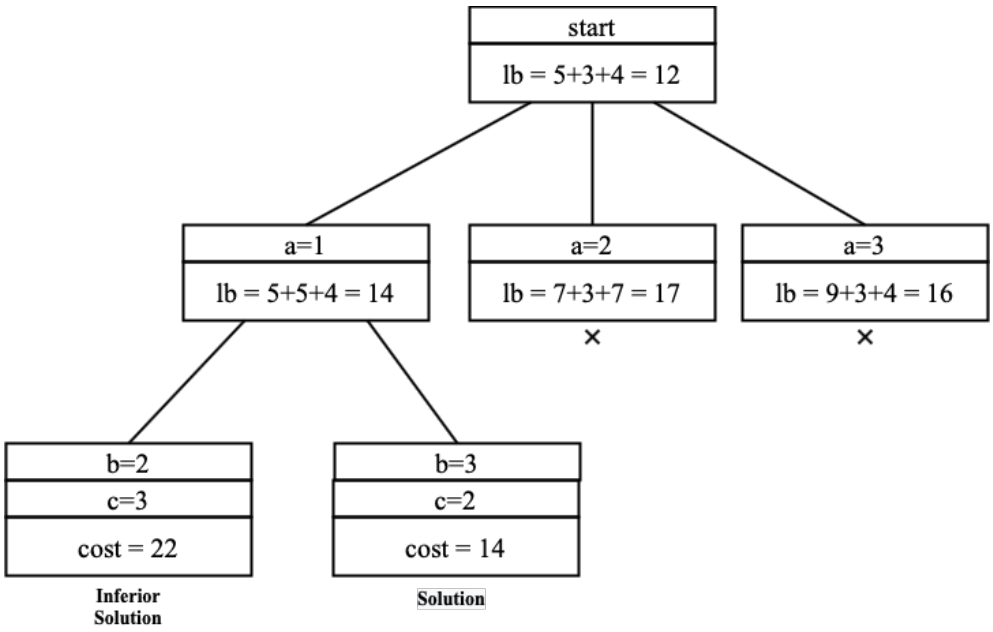

Figure 1: Sample solution to above instance of the assignment problem

Space-and-Time Trade-offs

1. Explain the theory behind space-and-time trade-offs.

This occurs where the running time of an algorithm is deemed as more important; thus, extra space is often used to reduce running time.

2. The following relates to the comparison counting sort algorithm.

   (a) What is the main theory behind the algorithm?

   The algorithm counts the total number of elements smaller than each element in order to determine an element's sorted index.

   (b) State the algorithm.

```
ComparisonCountingSort(A[0...n - 1])
//Sorts an array by comparison counting
//Input: An array A[0...n - 1] of orderable elements
//Output: Array S[0..n - 1] of A's elements sorted in non-decreasing order
for i = 0 to n - 1 do
    Count[i] = 0
for i = 0 to n - 2 do
    for j = i + 1 to n - 1 do
        if A[i] < A[j]
            Count[j] = Count[j] + 1
        else Count[i] = Count[i] + 1
for i = 0 to n - 1 do
    S[Count[i]] = A[i]
return S
```

   (c) Sort the given array $A[0...5] = \{34, 67, 12, 56, 14, 65\}$, where $n = 6$ using the comparison counting sort algorithm.

A[0...5]	34	67	12	56	14	65
Init. C[0...5]	0	0	0	0	0	0
i=0, C[0...5]	**2**	**1**	0	**1**	0	**1**
i=1, C[0...5]		**5**	0	1	0	1
i=2, C[0...5]			0	**2**	**1**	**2**
i=3, C[0...5]				**3**	1	**3**
i=4, C[0...5]					1	**4**
Fin. C[0...5]	2	5	0	3	1	4
S[0...5]	12	14	34	56	65	67

Table 2: A small instance of the comparison counting sort problem

3. The following relates to the distribution counting sort algorithm.

   (a) What is the main theory behind the algorithm?

   The algorithm works on the premise that we cannot overwrite any of the unsorted array's elements, whose values are between a lower bound l and an upper bound u. It copies the lowest value into a solutions array, such that it takes up the first f positions, where f is how many times it occurs in the original array. This is repeated for each new lower bound until the solutions array is full.

   (b) State the algorithm.

```
DistributionCountingSort(A[0..n-1],l,u)
//Sorts an array of integers from a limited range by distribution counting
//Input: An array A[0..n-1] of integers between l and u (l<=u)
//Output: Array S[0..n-1] of A's elements sorted in non-decreasing order
for j = 0 to u-l do
    D[j] = 0 //initialize frequencies
for i=0 to n-1 do
    D[A[i]-l] = D[A[i]-l]+1 //compute frequencies
for j=1 to u-l do
    D[j]=D[j-1]+D[j]//reuse frequencies for distribution
for i=n-1 downto 0 do
    j=A[i]-l
    S[D[j]-1]=A[i]
    D[j]=D[j]-1
return S
```

(c) Sort the given array $A[0...5] = 54, 37, 41, 37, 54, 5$, where $n = 6$ using the distribution counting sort algorithm.

Index	0	1	2	3
Array Values	5	37	41	54
Frequencies F[]	1	2	1	2
Distribution Values D[]	1	3	4	6

The resulting array is $S = \{5, 37, 37, 41, 54, 54\}$.