UWI

Design and Analysis of Algorithms

Tutorial 3 - Sample Solutions

Brute Force Approach

It is important we learn to calculate the time efficiencies for simple brute force algorithms and to also examine the theories behind them as well as how to work out simple examples of each.

1. Solve the travelling salesman problem using the brute force approach for the instance shown in figure 1 where the starting city is **a**.

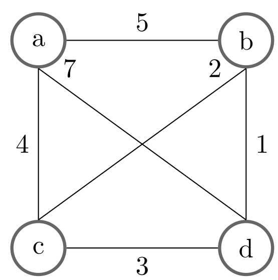

Figure 1: A small instance of the travelling salesman problem

To solve the travelling salesman problem, we must calculate the length of each possible route that can be taken, then we take the route path with the lowest cost.

Tour	Length	
a → b → c → d → a	5+2+3+7	17
a → b → d → c → a	5+1+3+4	**13***
a → c → b → d → a	4+2+1+7	14
a → c → d → b → a	4+3+1+5	**13***
a → d → c → b → a	7 +3+2+5	17
a → d → b → c → a	7+1+2+4	14

Table 1: Exhaustive search calculations for the travelling salesman problem

From table 1, observe that there are two optimal tours, both with a length of 13. These two tours are: a → b → d → c → a and a → c → d → b → a.

2. Solve the 0/1 knapsack problem using the brute force approach for the instance shown in table 2 with a max knapsack capacity of **12**.

Item	Weight	Value
1	5	$20
2	3	$15
3	9	$30
4	4	$25

Table 2: A small instance of the 0/1 knapsack problem

To solve the 0/1 knapsack problem, we must examine all of the possible combinations of items that there can be. We then determine which of those combinations can possibly fit in the knapsack and take the combination with the largest value.

Subset	Weight		Value	
$\emptyset$	0	0	0	\$0
{1}	5	5	20	\$20
{2}	3	3	15	\$15
{3}	9	9	30	\$30
{4}	4	4	25	\$25
{1,2}	5+3	8	20+15	\$35
{1,3}	5+9	14	-	-
{1,4}	5+4	9	20+25	\$45
{2,3}	3+9	12	15+30	\$45
{2,4}	3+4	7	15+25	\$40
{3,4}	9+4	13	-	-
{1,2,3}	5+3+9	17	-	-
{1,2,4}	5+3+4	12	20+15+30	**\$60***
{1,3,4}	5+9+4	18	-	-
{2,3,4}	3+9+4	16	-	-
{1,2,3,4}	3+5+9+4	21	-	-

Table 3: Exhaustive search calculations for the 0/1 knapsack problem

From table 3, observe that there is a single combination that yields the largest value and can fit into the knapsack. This combination features items 1, 2 and 4, which values sum to \$60 and weights sum to exactly 12.

3. Solve the assignment problem using the brute force approach for the instance shown in table 4.

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

Table 4: A small instance of the assignment problem

To solve the assignment problem, we must examine all the possible assignment combinations that each person can have (with the restriction of 1 job being assigned to exactly 1 person and vice versa). Then we determine the combination that yields the lowest value and select it as the optimal solution.

Assignment	Cost	
1,2,3	5+8+9	22
1,3,2	**5+5+4**	**14**
2,1,3	7+3+9	19
2,3,1	7+5+7	19
3,1,2	9+3+4	16
3,2,1	9+8+7	24

Table 5: A small instance of the assignment problem

From table 5, observe that the optimal combination yields a cost of 14. This combination features person 1 getting job 1, person two getting job 3 and person 3 getting job 2.

4. Analyse the following Bubble Sort algorithm and calculate its time efficiency using summation rules.

for $i \leftarrow 0$ to n–2 **do**
  for $j \leftarrow 0$ to $n-2-i$ **do**
    if $A[j] > A[j+1]$ **then**
      swap $A[j]$ and $A[j+1]$
    end if
  end for
end for

$$
\begin{align}
C(n) &= \sum_{i=0}^{n-2}\sum_{j=0}^{n-2-i} 1 = \sum_{i=0}^{n-2}[(n-2-i)-(0)+1] \tag{1}\\
&= \sum_{i=0}^{n-2}(n-1-i) = (n-1-0)\sum_{i=1}^{n-2}(n-1-i) \tag{2}\\
&= (n-1) + \sum_{i=1}^{n-2}(n-1-i) = (n-1) + \left[\sum_{i=1}^{n-2} n - \sum_{i=1}^{n-2} 1 - \sum_{i=1}^{n-2} i\right] \tag{3}\\
&= (n-1) + \left[n\sum_{i=1}^{n-2} 1 - \sum_{i=1}^{n-2} 1 - \sum_{i=1}^{n-2} i\right] \tag{4}\\
&= (n-1) + \left[n[(n-2)-(1)+1] - [(n-2)-(1)+1] - \sum_{i=1}^{n-2} i\right] \tag{5}\\
&= (n-1) + \left[n[n-2] - [n-2] - \sum_{i=1}^{n-2} i\right] \tag{6}\\
&= (n-1) + \left[n[n-2] - [n-2] - [\frac{(n-2)((n-2)+1)}{2}]\right] \tag{7}\\
&= (n-1) + \left[n^2 - 2n - n + 2 - [\frac{(n-2)(n-1)}{2}]\right] \tag{8}\\
&= (n-1) + \left[n^2 - 3n + 2 - [\frac{n^2 - n - 2n + 2}{2}]\right] \tag{9}\\
&= n - 1 + n^2 - 3n + 2 - \frac{n^2 - 3n + 2}{2} \tag{10}\\
&= n^2 - 2n - \frac{n^2 - 3n + 2}{2} + 1 \tag{11}\\
& \tag{12}
\end{align}
$$

Since $C(n) = n^2 - 2n - \frac{n^2-3n+2}{2} + 1$, we disregard constants an take the highest order of n, which implies that $C(n) \in \Theta(n^2)$.

5. Analyse the following Selection Sort algorithm and calculate its time efficiency using summation rules.

for $i \leftarrow 0$ to n–2 **do**
  $min \leftarrow i$
  for $j \leftarrow i+1$ to n–1 **do**
    if $A[j] < A[min]$ **then**
      $min \leftarrow j$
    end if
    swap $A[i]$ and $A[min]$
  end for
end for

$$
\begin{aligned}
C_w(n) &= \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2}[(n-1)-(i+1)+1] = \sum_{i=0}^{n-2}[(n-1-i-1+1] && (13)\\
&= \sum_{i=0}^{n-2}(n-1-i) = (n-1-0)\sum_{i=1}^{n-2}(n-1-i) && (14)\\
&= (n-1)+\sum_{i=1}^{n-2}(n-1-i) = (n-1)+\left[\sum_{i=1}^{n-2} n - \sum_{i=1}^{n-2} 1 - \sum_{i=1}^{n-2} i\right] && (15)\\
&= (n-1)+\left[n\sum_{i=1}^{n-2} 1 - \sum_{i=1}^{n-2} 1 - \sum_{i=1}^{n-2} i\right] && (16)\\
&= (n-1)+\left[n[(n-2)-(1)+1]-[(n-2)-(1)+1]-\sum_{i=1}^{n-2} i\right] && (17)\\
&= (n-1)+\left[n[n-2]-[n-2]-\sum_{i=1}^{n-2} i\right] && (18)\\
&= (n-1)+\left[n[n-2]-[n-2]-[\frac{(n-2)((n-2)+1)}{2}]\right] && (19)\\
&= (n-1)+\left[n^2-2n-n+2-[\frac{(n-2)(n-1)}{2}]\right] && (20)\\
&= (n-1)+\left[n^2-3n+2-[\frac{n^2-n-2n+2}{2}]\right] && (21)\\
&= n-1+n^2-3n+2-\frac{n^2-3n+2}{2} && (22)\\
&= n^2-2n-\frac{n^2-3n+2}{2}+1 && (23)\\
& && (24)
\end{aligned}
$$

Since $C(n) = n^2 - 2n - \frac{n^2-3n+2}{2} + 1$, we disregard constants an take the highest order of n, which implies that $C(n) \in \Theta(n^2)$.