Exercise 4.8

Cemal Arican

February 2020

1 The problem

Consider the following scheduling problem. Given that there are n jobs, each which needs to be processed on one of two machines. Each machine can process at most one job at a time. Processing job j takes p_j time units, independent of the machine which it is processed. The goal is to assign half of the jobs and that the maximum load of the two machines is minimized. Such that S_i denoted the set of jobs allocated to machine i. Moreover, $|S_1| = |S_2| = \frac{n}{2}$ with objective minimize max $\{\sum_{j\in S_1} p_j, \sum_{j\in S_2} p_j\}$. The difference between this problem and the partition problem, is that we are constrained such that the set created for each machine is equal to half the number of number of jobs.

2 The Dynamic Program

In dynamic program we first need to define the subproblem. Once we have defined the subproblem, we can solve it in three steps: Initialization, Recursion and the Output.

2.1 The Subproblem

Let $B = \frac{\sum_{j=1}^{n} p_j}{2}$, and let $v \in \{0,...,B\}$ and $k \in \{1,...,\frac{n}{2}\}$. Then let us consider the following binary function:

$$F(i,v,k) = \begin{cases} 1 & \text{if } \exists S_1 \in \{1,...,i\}, v \in \{0,...,B\}, k \in \{1,...,\frac{n}{2}\}, \\ 0 & \text{otherwise.} \end{cases}$$

2.2 Dynamic Program:

Algorithm 1: Initialization

```
for i ∈ {1,...,n} : do
    // O(n)
    for v ∈ {0,...,B} : do
        // O(B)
        for k ∈ {0,..., n/2} do
            // O(n/2)
            if v = p_1 or v = 0 or k = 0 : then
                F(i,v,k) = 1
            else
                F(i,v,k) = 0
            end
        end
    end
end
```

Algorithm 2: Recursion

```
for i ∈ {1,...,n} : do
    // O(n)
    for v ∈ {0,...,B} : do
        // O(B)
        for k ∈ {0,..., n/2} do
            // O(n/2)
            if p_i ≤ v then
                F(i,v,k) = F(i−1,v,k) or F(i−1,v−p_i,k−1)
            else
                F(i,v,k) = F(i−1,v,k)
            end
        end
    end
end
```

The output $F(n, v, \frac{n}{2}) = 1$. In other words, we want to make sure that the algorithm returns *true* whenever there is a set such that we can have half number of all possible jobs. In order to find the appropriate set, we would have to backtrack.

Total Running Time:

Looking at all the parts of the algorithm, we have a Total Running Time, $RT(n) = \mathcal{O}(n \times B \times \frac{n}{2} + n \times B \times \frac{n}{2} + 1) \in \mathcal{O}(n^2 B)$.

Polynomial Input size?

Let $W = max_{\forall j \in \{1,\ldots,n\}}\{p_i\}$. Then the input size is $< I > \in \mathcal{O}(n + nlog(W))$, i.e. the sum of the array length and the time need to program each value in the array. The algorithm is pseudo-polynomial as the $W \in B$. Meaning that the value of W increases, so does B and in tandem the algorithm running time increases. Hence, the algorithm is pseudo-polynomial as the running time of the algorithm is also dependent in the input size and their individual values.