# Assignment 3

Quinn Perfetto, 104026025
60-454 Design and Analysis of Algorithms

March 19, 2017

**Question 1 (a).**

**Idea:** Sum consecutive elements of the input array until the sum exceeds $M$. Once this happens, add the offending index to the subdivision and reset the sum.

```
Algorithm 1: Subdivide(W, M)
  Input: $W[1..n], 0 \leq W[i] \leq M, 1 \leq i \leq n$
  Output: $S[1..k]$ such that $S$ is an optimal subdivision of $W$
  begin
      sum := 0;
      S := [ ];
      for $i \leftarrow 1$ to $n$ do
          sum = sum + $W[i]$;
          if $sum > M$ then
              append(S, i - 1);
              sum := $W[i]$;
          end
      end
  end
```

**Lemma 1.1.** Algorithm Subdivide produces a valid subdivision of the input array $W$

We shall show this by inductively proving that after the m$th$ iteration of the for loop,

$$S \text{ is a valid subdivision of } W[1..m] \wedge sum = \sum_{j=S_{last}+1}^{m} W[j]$$

**Note:** We take $S_{last}$ to be the last element in $S$ if it exists, and 0 otherwise.

*Proof.* (Induction Basis) We first note that sum is initialized to 0. After control reaches line 4 for the first time we have,

$$sum = sum + W[1] \Rightarrow sum = W[1] = \sum_{j=1}^{1} W[j]$$

Note that $S$ was initialized to [ ]. Since $sum = W[1] \leq M$, control will not enter the if statement on line 5, thus $S$ will remain empty and $S_{last} = 0$. Further since $W[1..m = 1]$ is a single element list such that $W[1] \leq M$, $S =$ [ ] is vacuously a valid subdivision of $W$.

(Induction Hypothesis) Assume that after $k$ iterations of the for loop,

$$S \text{ is a valid subdivision of } W[1..k] \wedge sum = \sum_{j=S_{last}+1}^{k} W[j]$$

(Induction Step) **Case 1:** $sum > M$

By the induction assumption $S$ is a valid subdivision of $W[1..k]$, by the defintion of a valid subdivision we thus have,

$$\sum_{j=S_{last}+1}^{k} W[j] \leq M \qquad \text{(I)}$$

After appending $i - 1 = k$ to $S$, $S_{last} = k$. Therefore (I) is equivalent to,

$$\sum_{j=S_{last-1}+1}^{S_{last}} W[j] \leq M$$

Further since $\sum_{j=S_{last}+1}^{k+1} W[j] = W[k+1] \leq M$ we have $S$ is a valid subdivision of $W[1..k+1]$.

After assigning $sum = W[k+1]$ we also have $sum = \sum_{S_{last}+1}^{k+1} W[j]$.

**Case 2:** $sum \leq M$

Since by our inductive assumption $S$ is a valid subdivision of $W[1..k]$ and,

$$\begin{aligned} sum &= \sum_{j=S_{last}+1}^{k} W[j] + W[k+1] \\ &= \sum_{j=S_{last}+1}^{k+1} W[j] \\ &\leq M \end{aligned}$$

We have S is a valid subdivision of $W[1..k+1]$. □

Therefore by Lemma 1.1, after $n$ iterations $S$ will be a valid parition of $W[1..n]$.
Hence the algorithm produces a valid subdivision of $W$.

**Lemma 1.2.** Algorithm Subdivide produces an optimal subdivision of the input array in terms of size

*Proof.* (Contradiction) Suppose to the contrary that Algorithm Subdivide does not produce an optimal subdivision of the input array. Let $S$ be the subdivision produced by Algorithm Subdivide for some input array $W$, and let $S'$ be a valid subdivision of $W$ such that $|S'| < |S|$ (i.e. $S'$ is more optimal than $S$). Since $|S'| < |S|$, $\exists i_j, i_{j+1} \in S$ and $\exists i_x, i_{x+1} \in S'$ such that $i_x \leq i_j < i_{j+1} < i_{x+1}$ (I). Such indices must exist for if they didn't the paritions would be equal. It can be seen that extending any subdivision produced by Algorithm Subdivide would result in an invalid subdivision as,

$$\exists i_k, \sum_{j=i_k+1}^{i_{k+1}} W[j] > M$$

(I) implies that $S'$ contains a subdivision that is an extension of a subdivision of $S$, and thus has a sum larger than $M$. Therefore $S'$ is an invalid subdivision. Thus $S$ must be optimal. □

**Lemma 1.3.** Algorithm Subdivide runs in O(n) time

The for loop iterates over each of the $n$ elements of $W$, and performs a constant amount of work in each iteration. We therefore have T(n) = O(cn) = O(n).

**Question 1 (b).** The greedy algorithm presented above will not produce an optimal subdivision if $W$ contains negative elements. If $W$ contains negative numbers extending a subdivision does not necessarily increase it's sum, and thus greedily ending subdivisions does not guarantee optimality.

**Example:** W $= [1, 2, 10, -9]$, $M = 10$
$S_{greedy} = [2]$, $S_{optimal} = []$

**Question 2 (a).**

**Question 2 (b).**

**Idea:** Let $D[i, j]$ represent the least difference matching between $H[i..n]$ and $S[j..m]$. For any $i$, $j$ two options arise:

- Pair $H_i$ with $S_j$, in which case $D[i,\ j] = |H_i - S_j| + D[i+1, j+1]$
- Don't pair $H_i$ with $S_j$, in which case $D[i,\ j] = D[i, j+1]$

The optimal result is thus the minimum of the two options.

**Algorithm 2:** LeastDifferenceMatching(H, S)

```
Input: $H = \{h_j \mid 1 \leq j \leq n\}, S = \{S_j \mid 1 \leq j \leq m\},\ n \leq m$
Output: $min(\sum_{i=0}^{n} |H[i] - S[i]|)$
begin
    for $i \leftarrow 1$ to $m$ do
        $D[n+1,\ i] = 0$;
    end
    for $i \leftarrow 1$ to $n-1$ do
        $D[i,\ m] = \infty$;
    end
    Sort(H);
    Sort(S);
    $D[n,m] = |H[n] - S[m]|$ ;
    for $i \leftarrow n$ to $1$ do
        for $j \leftarrow m$ - $1$ to $1$ do
            $D[i,j] = \min(|H[i] - S[j]| + D[i+1,\ j+1],\ D[i,\ j+1])$;
        end
    end
    return $D[1,1]$
end
```