Brandon Toner
September 16, 2021

CS 405: Algorithm Analysis II
Homework 7: Greedy Algorithms

1. The Nth floating point number will either be in one of the intervals found from processing the first $N-1$ numbers, or a new interval. This algorithm assumes that if $G_N \in [v.max - 1, v.min + 1]$ then G_N is in one of the previous intervals, and there is no need to check any more intervals.

Algorithm 1: UnitInterval

```
Data: G, N
begin
    if N = 1 then
        return (min = G_0, max = G_0)
    else
        data = UnitIntervals(G, N − 1)
        for v ∈ data do
            if G_N ≤ v.min + 1 and G_N ≥ g.max − 1 then
                if G_N > v.max then
                    v.max = G_N
                if G_N < v.min then
                    v.min = G_N
                return data
        return data + (min = G_N, max = G_N)
```

2. This algorithm assumes that since multiplication is commutitiave, we can rearrange the product in a way that the fist element in the product will be the largest. Therefore the greedy choice for the first value will be to choose i, j such that ${a_i}^{b_j}$ is a maximum. This value will be $a_m^{b_m}$ where $a_m = max(A)$ and $b_m = max(B)$, since decreaing the exponent or the base would decrease the value. As for the optimal substructure, if we assume we have a perumtation, S that is the optimal solution, and a subproblem $p \subset S$ p must also be an optimal solution. If there were another permutation of the exponents that yeilded a larger product, we would be able to repalce the elements of p in S to make a larger solution, which is a contradiction that S is the optimal solution.

Algorithm 2: MaxProduct

```
Data: A, B
begin
    if len(A) = 1 then
        return pow(a_0, b_0)
    else
        a_m = max(A)
        b_m = max(B)
        return pow(a_m, b_m) ∗ MaxProduct(A − a_m, B − b_m)
```