Assignment Five

Josh Seligman

joshua.seligman1@marist.edu

November 17, 2022

1 Fractional Knapsack Algorithm

1.1 The Algorithm

1.2 Asymptotic Analysis

2 Appendix

2.1 Fractional Knapsack Algorithm

```
1  void runAlgo(SpiceArr* spices, Queue<int>* knapsacks) {
2      // Start off by running a sort on the spices array to make them in descending order
3      quickSort(spices);
4
5      while (!knapsacks->isEmpty()) {
6          Node<int>* curKnapsack = knapsacks->dequeue();
7
8          // Create an array that corresponds with the spice array for keeping track of what
               was taken by the knapsack
9          int quantityTaken[spices->length];
10         for (int i = 0; i < spices->length; i++) {
11             quantityTaken[i] = 0;
12         }
13
14         // Start off with an empty knapsack and a value of 0
15         int capacityLeft = curKnapsack->data;
16         double spiceValue = 0;
17
18         // Start considering the first element in the array (most valuable per unit)
19         int spiceIndex = 0;
20
21         // Continue until the knapsack is full or until there is no more spice to take
22         while (capacityLeft > 0 && spiceIndex < spices->length) {
23             // If there is space for the entire pile of spice, take it all
24             if (capacityLeft >= spices->arr[spiceIndex]->getQuantity()) {
25                 // Update the array of spice taken
26                 quantityTaken[spiceIndex] = spices->arr[spiceIndex]->getQuantity();
27
28                 // Be greedy and take everything available if possible
```

```
29                  capacityLeft -= spices->arr[spiceIndex]->getQuantity();
30                  spiceValue += spices->arr[spiceIndex]->getPrice();
31              } else {
32                  // Update the table entry
33                  quantityTaken[spiceIndex] = capacityLeft;
34
35                  // Compute the value of the spice we can take
36                  spiceValue += capacityLeft * spices->arr[spiceIndex]->getUnitPrice();
37
38                  // Update the capacity to be 0
39                  capacityLeft = 0;
40              }
41              // Go on to the next spice
42              spiceIndex++;
43          }
44
45          // Start with this text
46          std::cout << "Knapsack of capacity " << curKnapsack->data << " is worth " <<
                spiceValue << " quatloos and contains";
47
48          // Iterate through all of the spices
49          for (int j = 0; j < spices->length; j++) {
50              // Only print out the spices we take
51              if (quantityTaken[j] > 0) {
52                  // Little formatting logic
53                  if (j > 0) {
54                      std::cout << ", ";
55                  } else {
56                      std::cout << " ";
57                  }
58                  // The amount and name of the spice taken
59                  std::cout << quantityTaken[j] << " scoops of " << spices->arr[j]->getName();
60              }
61          }
62          std::cout << "." << std::endl;
63
64          // Memory management
65          delete curKnapsack;
66      }
67  }
```

Listing 1: Fractional Knapsack Algorithm (C++)