Stephen Andrews

CS 2223

Project 2

April 24th, 2016

Exhaustive vs. Greedy Algorithms

Executive Summary:

Exhaustive search and greedy algorithms drastically differ in their implementations, but can at times yield the same final solution. An exhaustive search systematically enumerates all possible solutions for a problem and chooses the globally optimal solution, whereas a greedy algorithm makes a locally optimal choice at each stage during computation with the hope of finding the globally optimal solution.

Pseudocode:

```
1  def exhaustive(matrix):
2      p = permutate(matrix_size)
3      lowest = -1
4      assignment = []
5      for i in p:
6          index_assignment = p[i]
7          cost = 0
8          for j in p[0]:
9              cost += matrix[j][index_assignment[j]]
10         if cost < lowest or lowest == -1:
11             lowest = cost
12             assignment = index_assignment
13     return assignment
```

```python
def greedy(matrix):
    n = len(matrix)
    assignment = []
    for p in n:
        person = matrix[p]
        lowest = -1
        for c in n:
            cost = person[c]
            if (cost < lowest or lowest == -1) and not (c in assignment):
                lowest = cost
                assignment[p] = c
        lowest = -1
    return assignment
```

Questions:

---

1. The time efficiency of the greedy algorithm is $O(n^2)$ since there exists a loop nested within a loop. The brute force implementation has a much greater time complexity of $\Theta(n!)$ since for every solution we must generate a permutation of the n messages where we have n people available to decrypt n messages.
2. The greedy implementation definitely works better for larger inputs in terms of time efficiency but not necessarily complete correctness.
3. The greedy algorithm does not always yield the optimal solution for an n x m matrix. In fact, the sample data given in the assignment is a good example of how the greedy implementation may not always reach a globally optimal solution. The globally optimal solution yields a total cost of 13, while the greedy implementation yields a total cost of 14. The following tables show the total cost of executing both algorithms on the example data.

Exhaustive (13)

Name	Message	Cost
Jill	2	2
Sven	1	6
Bud	3	1
Kevin	4	4

Greedy (14)

Name	Message	Cost
Jill	2	2
Sven	3	3
Bud	1	5
Kevin	4	4