hw 6

jack frate[1]

[1]Rochester Institute of Technology

December 7, 2018

1. The algorithm is greedy, as it looks for the best choice at each step, instead of overall. It is the same problem as the original, just in reverse. This is optimal because even though the list is reversed, it is still a greedy algorithm and will return an optimal result.

2.

In the below example, selecting the activity of least duration is not optimal.

i	1	2	3
si	1	3	4
fi	4	5	8

In this example, we would choose column 2, as it has the shortest duration, where the best solution would choose to use columns 1 and 3.

i	1	2	3	4	5	6	7	8	9	10
si	1	2	2	2	3	4	5	6	6	6
fi	3	4	4	4	5	6	7	8	8	8
#	3	4	4	4	4	2	4	4	4	4

In this example, when we select the activity with the least amount of overlap would result in selecting column 6, then it would have to select column 11 or 1. The result would be an activity list length 3, of columns 1, 6, and 11. The optimal choice to make in this scenario is an activity list length 4, of columns 1, 5, 7, and 11.

i	1	2	3	4
si	1	3	4	0
fi	4	5	8	8

In this example, the algorithm that selects the earliest start time would select column 4, where the optimal solution would be to select columns 1 and 3.

3.

An optimal solution to the knapsack problem is the one with the highest total value density. As we always add the highest value that we can, we will eventually get to the highest value density. This is becuase it is greedy, and will always pick the item with the most value.

4.

With G1 and G2, we can assume that both have n vertices, since isomorphic graphs have the same amount of vertices.

To determine of any two graphs are isomorphic,

1. Compute every possible permutation of G1, storing each permutation as a certificate.

2. Compare each certificate to G2, if they are the same, G1 and G2 are isomorphic, if there is no such certificate, they are not.

5.

If we assume the contrapositive,P = NP therefor NP = coNP

if P = NP, then both

a. for every element el in NP, we have el in P, and since the p is closed off by compliment, compliment of el is in P, and as a result el is in coNP.

b. for each el in coNP, we get the compliment of el in P, and following the same rule of compliments for P, we get el in NP.

6.

// apologies for the latex formatting not working, but you should be able to infer what I mean.

$Lety = ((x_1[?]x_2)[?]x_3)[?]((x_1[?]x_2[?]!x_3)[?]x_3)[?](x_1[?]x_2[?]!x_3)$

$LetA = ((x_1[?]x_2)[?]x_3)$

$LetB = ((x_1[?]x_2[?]!x_3)[?]x_3)$

$LetC = (x_1[?]x_2[?]!x_3)$

We need A, B, and C to be true in order to satisfy our problem.

For C to be true, we need x3 to be false and x1, x2 need to be true. If we put these values in the equation, the following will happen:

$y = ((x_1[?]x_2)[?]x_3)[?]((x_1[?]x_2[?]!x_3)[?]x_3)[?](x_1[?]x_2[?]!x_3)$

$y = ((1[?]1)[?]0)[?]((1[?]1[?]!0)[?]0)[?](1[?]1[?]!0$

$y = ((1[?]1)[?]0)[?]((1[?]1[?]1)[?]0)[?](1[?]1[?]1)$

$y = (1[?]0)[?](1[?]0)[?]1$

$y = 0[?]1[?]1$

$= 0$

Because this leads to false, we can determine that y is not satisfiable.

7.

Determining the satisfiability of a formula in disjunctive normal form is making a formula where there are multiple sub formulas containing 'and' and'or' clauses. These will only work if they will evaluate to 1. Since the variables are either true or false (1 or 0), all the possible combinations are n^2. In order to check these, we have to check each subformula. This means that there will be n problems, with n solutions each. So it would take n^2 tries at most to verify the formula.

8.

a.

```
Knapsack(W, i, w, v)
    If (W == 0) return 0
    elIf (n == 0) return 0
    elIf (w[i-1] > W) return Knapsack(W, i-1, w, v)
    elIf (w[i-1] [?] W) return selectMax(
                        Knapsack(W, i-1, w, v),
                      Knapsack(W-w[i-1], i-1, w, v) + v[i-1]
)
  selectMax(a, b)
      if(a > b) return a
      elIf(b [?] a) return b
```

c.

n = amount of items

W = max waight of knapsack

time complexity = O(nW)

d.

I could not prove that P = NP. This is because the solution is not actually polynomial, but rather pseudo polynomial. Because of the way that the algorithm works (nested loops from n and W), the amount of time it takes to process the solution will increase exponentially.