Hazal Gönen

CSE-321 - Introduction to Algorithm Design

Homework 05 26.12.2016

1.) Answer question below.

a) Why do we use greedy algorithms?

b.) Do greedy algorithms always fail to find globally optimal solution? Why or whynot? Provide concrete examples.

a) Greedy algoritması açgözlü algoritma olarak da adlandırılmaktadır. Buradan da anlaşılacağı üzere algoritma mümkün olan uç sonuca en yakın olan seçimi yaparak ilerler.
En büyük avantajlarından biri hızlı olmasıdır.

b.) Greedy algoritma genel problemi daha küçük parçaları çözerek başlar. Parçalanabilen en küçük parçanın optimum çözümünü hesoplarsak onu referans alarak bir büyük parçanın da optimuma yakın çözümünü bulabiliriz. Local olarak optimum olan çözüm global olarak her zaman optimum olmayabilir.
Örneğin elimde 14 T para birimim var ve T, 5T, 7T, 10T bozuk paralarım var. Greedy algoritma 14'e en yakın maksimum bozuk parayı verip kalanı da optimum şekilde verir. Yani

14 T → 10 T + T + T + T + T ⇒ Toplam 5 tane para kullanıldı

14 T → 7 T + 7 T ⇒ 2 tane para kullanıldı.

Burada global olarak optimum çözümü bulamadığımızı kanıtlamış olduk.

2.) Prove or disprove the following statements.

a.) If e is a minimum-weight edge in a connected weight graph, it must be among edges of at least one minimum spanning tree of the graph.

b.) If e is a minimum-weight edge in a connected weight graph, it must be among edges of each minimum spanning tree of the graph.

c.) If edge weights of a connected weighted graph are all distinct, the graph must have exactly one minimum spanning tree.

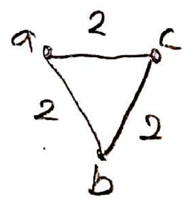
d.) If edge weights of a connected weighted graph are not all distinct, the graph must have exactly one minimum spanning tree.

Kruskal algoritması ağırlıklı bir graphda bütün vertexleri içeren en kısa yolu bulur.

1. n kenarlı bir graph için herhangi bir düğüm seçilir.
2. Buna en yakın olan ve döngü oluşturmayan diğer vertex eklenir.
3. n-1 kenar eklenene kadar devam edilir.

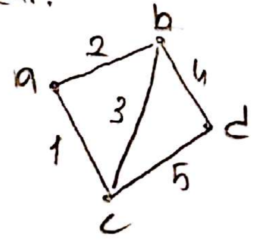
a.) Algoritma bütün düğümleri gezdiği için ve her düğümün kendisine en yakın (minimum-weight) düğümü seçtiği için ifade kesinlikle doğrudur.

b.) Eğer minimum weight 2 ise ve graph yanda ki gibiyse a bc seçildiği taktirde a-c edge'i minimum weight olmasına rağmen seçilmemiş olacaktır. Yanlış bir ifadedir.



b ∪ a - c - d

c.) Eğer bütün ağırlıklar farklı ise exactly 1 minimum spanning tree'si vardır.



örneğin a seçildi.	a
a'ya en yakın c var	a - c
c'ye en yakın b var	a - c - b
b'ye en yakın d var	a - c - b - d

d.) Kesinlikle yanlıştır. b şıkkında farklı ağırlıkta olmayan graph çizmiştim.
a-b-c , a-c-b , c-b-a bunların hepsi minimum spanning treedir.

3. Design a greedy algorithm to solve 0/1 Knapsack Problem. Describe your greedy approach in detail. Write a program to show your algorithm including test on a sample set. Write code in Python. Analyze its best case, worst case, and average case complexities.

Bu problem bir torbanın icerisine en fazla esyanın yerlestirilmesini hedefler.

Agırlık olarak az, mali degeri cok olan esyalar secilmelidir.

Esya ya alınır ya bırakılır

W => kapasite

$$\text{knapsack\_items} [(\overset{\uparrow \text{ index}}{} , \overset{\uparrow \text{ weight}}{} , \overset{\nearrow \text{ value}}{}) \Rightarrow$$ bütün bilgileri tutan liste

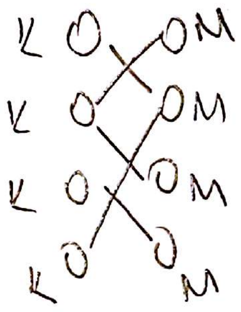
- ilk olarak bütün listeyi sıralarım (maliyetine göre) (büyükten küçüğe)
- Listenin elemanları bitene kadar ya da kapasiteye ulasana kadar devam ederim.
- Eger itemin degeri ve cantanın agırlıgının toplamı kapasiteyi gecmezse
  cantaya o itemi koyarım
  cantanın agırlıgını güncellerim
  cantanın degerini güncellerim.
- Kapasite gecilirse atlarım.

5. Design a greedy algorithm to solve Map Coloring Problem. Describe your greedy approach in detail. Write a program to show your algorithm including test on a sample set. Write code in Python. Analyze its best, worst, average case complexities.

Harita boyama algoritmasında haritayı en az renkle boyamak ve komşuları farklı renkle boyamak asıl amaçtır.

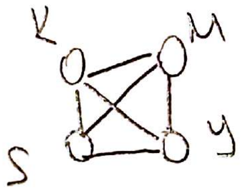
Algoritması : • ilk vertexe renk ata (örn kırmızı)
- 2. vertex ilk vertexin komşusu ise farklı renk ata (mavi)
- 2. vertex ilk vertexin komşusu değilse aynı renk ata (kırmızı)
- n. vertexe komşuları boyasızsa ilk rengi ata (kırmızı)
- n. vertexe komşuları boyalıysa boyasız dan ilk rengi ata

Algoritma en iyi bipartite graphlarda çalışır.



⇒ 2 renk kullanarak bütün harita boyadı.

Bütün graphların komşu olduğu durumda en kötü çalışır.



6. Consider the problem of scheduling n jobs of known durations $t_1, \dots t_n$ for execution by a single processor. The jobs can be executed in any order, one job at a time. You want to find a schedule that minimizes the total time spent by all the jobs in the system.

Design a greedy algorithm for this problem. Does the greedy algorithm always yield an optimal solution? Prove or disprove

işleri nondecreasing olarak sıralanmalıdır.

toplamsüre $= t_{i_1} + (t_{i_1} + t_{i_2}) + \dots + (t_{i_1} + t_{i_2} + \dots t_{i_n}) = n t_{i_1} + (n-1) t_{i_2} + \dots + t_{i_n}$

Toplamı en aza indirmek için daha küçük t'leri daha büyük sayılara atamalıyız.

$i_1, i_2, i_3 \dots i_n$ which $t_{i_k} > t_{i_{k+1}}$

↳ sistemdeki toplam süre azaltılabilir.

$$\left[\left(\sum_{j=1}^{k-1} t_{i_j} + t_{i_{k+1}} \right) + \sum_{j=1}^{k-1} t_{i_j} + t_{i_{k+1}} + t_{i_k}) \right] -$$

$$\left[\left(\sum_{j=1}^{k-1} t_{i_j} + t_{i_k} \right) + \left(\sum_{j=1}^{k-1} t_{i_j} + t_{i_k} + t_{i_{k+1}} \right) \right]$$

7. Design a Greedy algorithm for the assignment problem (see section 3.4) Does your greedy algorithm always yield an optimal solution? Prove or disprove.

Atama problemlerinin amacı ise en uygun kişiyi atayarak en düşük maliyetli atamayı sağlamaktır.

Algoritma : . Matrixde bulunan en küçük elemanı seç. Satır sütun işaretle (5 seçildi 1. sütun 3. satır boyandı)

$$\begin{bmatrix} \cancel{20} & \cancel{30} & \cancel{40} \\ \cancel{10} & \cancel{20} & \cancel{25} \\ \textcircled{5} & \cancel{50} & \cancel{80} \end{bmatrix}$$

55

• Kalan matrixde en küçüğü seç, boya (20 seçildi)

(5,20,40) optimum çözüm değildir.

(10,20,25) optimum çözümdür.

Greedy algoritması her zaman optimum çözümü vermez.

8.) Write a pseudocodo of the greedy algorithm for the chage-making problem, with an amant n and coin denominations $d_1 > d_2 > \ldots > d_m$ as its input. what is the time efficiency class of yaur algorithm.

Algorithm:

```
                 toplampara         bozuk para
                      ↑           ↗
Chage Making ( n, d[1..m])
      int C[];
      for i←1 to m do
            if n=0
                return C
            C[i] = floor (n/d[i])
              n = n % d[i]
return null.
```

return null

Algoritmanın time efficiency: $\sum_{i=1}^{m} c = m \cdot \cancel{c} = O(m)$

toplam parayı bozukluklarla ifade etme problemi şu şekildedir:

$n = 50$ $\quad d[1] = 20 \quad C[1] = 50/20 = 2$

$\qquad\qquad\qquad\qquad\qquad 50 \% 20 = 10$

$d[2] = 10 \quad C[2] = 10/10 = 1$

$d[3] = 5 \qquad\qquad 10 \% 10 = 0 \Rightarrow$ if koşulu

$d[4] = 1$

Sonuç olarak 50 lirayı $C[1] = 2$ tane 20, $C[2] = 1$ tane 10 } lirayla ifade ettim.

9. a. How can we use Prim's algorithm to find a spanning tree of a connected graph with no edges?

b. Is it a good algorithm for this problem?

a) Prim algoritmasının calısma mantığı ağırlıklara dayanmaktadır. Dolayısıyla eger ağırlığın bir önemi yok ise bu algoritmayı kullanabilmek için sonucu degistirmeyeceğinden 1 ağırlık verebiliriz.

b.) Travel algoritmalarından depth-first search yada breadth first search algoritması kullanılabilir.