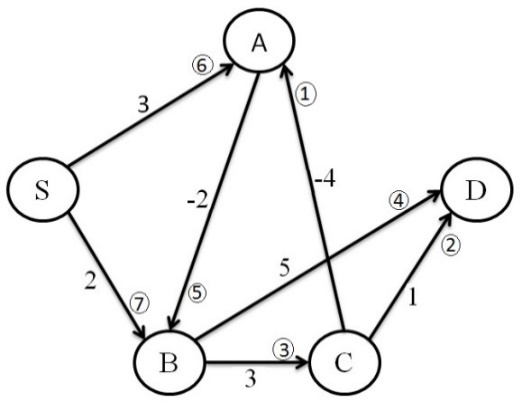
**BME222 ALGORİTMALAR BÜTÜNLEME SINAVI**

|  |  |
| --- | --- |
| **Q1. (a)** Aşağıdaki fonksiyonu, New(5,1) ilk çağrısı ile çalıştırırsak bu fonksiyon bize neyi hesaplar? **[10P]**  function New(n,acc)  if n == 1  return 1\*acc;  end  return New(n-1,acc\*n);  end | **(b)** Yandaki recursive fonksiyonu iteratif fonksiyona çeviriniz. **[10P]** |

**Q2.** Aşağıda verilen yönlü graf üzerinde **S** düğümünü başlangıç olarak kullanarak Bellman-Ford Algoritmasını çalıştırınız. Çıktı olarak ne elde edilir? Açıklayınız. Not: Ayrıt numaraları daire içine alınarak belirtilmiştir. Ağaç çizmeyi unutmayınız. **[20P]**



**Q3. I.** Aşağıdaki ifadelerin doğru veya yanlış olduğunu belirtmek için D veya Y’yi daire içine alın. **(8P)**

1. **D Y** Bellman Ford algoritmasının zaman karmaşıklığı O(|V|+|E|) dir.
2. **D Y** Dijkstra’s Algoritması bir Böl ve Fethet Algoritması değildir.
3. **D Y** Özyinelemeli (Rekürsif) algoritmalar her zaman iteratif algoritmalardan daha iyidir.
4. **D Y** Böl ve Fethet algoritmalarında özyinelemeli prosedürler uygulanır.
5. **D Y** Ağırlıklı bir grafta kenar ağırlıklarının tamamı birbirinden farklı ise, minimum kapsayan ağaç her zaman tektir.
6. **D Y**
7. **D Y** Flooyd-Warshall algoritması dinamik programlama örneği değildir.
8. **D Y** Çabuk sıralamanın en iyi durumu her bölütleme (partition) adımında diziyi ortadan ikiye böldüğü durumdur.

**II.** Aşağıdaki ifadeleri tamamlayınız **(12P)**

1. Merge sıralamasının çalışma zamanı …………………. dır.
2. ………….. sıralamasının en kötü durum çalışma zamanı dır.
3. Strassen algoritması 8 adet özyinelemeli çarpma yerine …. adet özyinelemeli çarpma kullanır.
4. İkili arama ağaçlarında yapılan işlemlerden üç tanesi …………………………………………………..dır.
5. Dinamik programlama çözdüğü altproblemleri ……………………… saklar.
6. Negatif ağırlıklı graflarda en kısa yolları bulmak için ………………………… algoritması kullanılır.

**Q4.** {37,61,12,83,25,22,61,93} verilen dizi için, aşağıda verilen TREE-INSERT algoritmasını kullanarak sayıları sırasıyla boş bir ikili ağaca ekleyiniz. **preorder** gezinme yöntemi ile ağacı dolaşın ve çıktıyı yazınız. Tek bir ağaç çiziniz. Her eklemede yeni bir ağaç çizmeyiniz. **[20P]**

TREE-INSERT(key,root){

if(root==NULL){

newnode.data=key;

root=newnode;

}

else{

if(key<=root->data)

TREE-INSERT(key,root.left);

else

TREE-INSERT(key,root.right);

}

return root;

}

**Q5.** Verilen bir n sayısı için n. Fibonacci sayısını bulan bir algoritmayı dinamik programlama ile yazınız. **[20P]**

|  |  |
| --- | --- |
| **Q6.** AşağıdaDijkstra algoritması ve ağırlıklı yönlü bir graf verilmiştir; | |
|  |  |
| https://www.cs.bham.ac.uk/~mdr/teaching/modules04/java2/Dijkstra.gif |  |
| 1. A kaynak düğümünden diğer düğümlere giden en kısa yolun yolların maliyetlerini bulunuz. **(**Direkt çözümler kabul edilmeyecektir. Tablo kullanmanız tavsiye edilir.) **[15P]** 2. Veri yapısı olarak Fibonacci heap i kullanırsak Dijkstra algoritmasının çalışma zamanı ne olur? **[5P]**  |  |  |  |  | | --- | --- | --- | --- | | **Data Type** |  |  | **TOTAL TIME** | | Fibonacci\_Heap |  |  | ? | | |