Dugga i Objektorienterad modellering och design (OMD) Helsingborg – grafalgoritmer

EDAF25

21 mars 2025, 10-12

- Skriv ditt namn och personnummer på avsedd plats nedan på denna sida, *men inte på några andra sidor*.
- Ha legitimation i beredskap för id-kontroll.
- Skriv i första hand dina svar på frågorna på avsedd plats i detta häfte.
- Skriv bara på framsidan av arken.
- Behöver du mer plats kan du använda det blanka arket i slutet av häftet eller, om det inte räcker, extra tomma ark. Se då till att märka dem med din anonymkod.
- Skriv tydligt – om texten inte går att läsa kan du inte få några poäng.

Inlämning

- På baksidan av varje ark i detta häfte finns en anonymkod angiven.
- Skriv din anonymkod på baksidan av varje *extra ark* om du lämnar in sådana.
- Om du lämnar in extra ark, eller har tagit isär arken i häftet, så använd häftapparaten som finns tillgänglig för att häfta ihop arken.
- Ha legitimation i beredskap om id-kontroll inte redan hunnit utföras.
- Lämna svarshäftet till tentamensvakten som kommer att riva av första sidan med namn och personnummer i svarshäftet. Denna kommer förvaras separat tills dess alla duggor är rättade och först då användas för att identifiera den skrivande.

Hjälpmedel: • Java snabbreferens

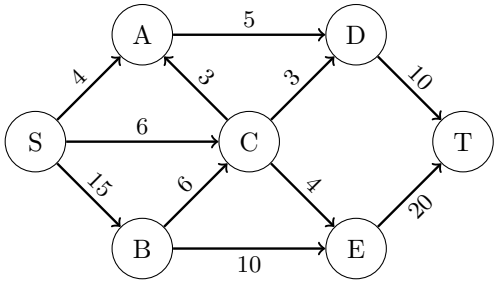
Namn:

Personnummer: *Sign. ID-kontroll:*

Uppgift 1

Om man vill räkna ut det maximala flödet genom en graf kan man använda sig av Ford-Fulkersons metod. Denna metod går ut på att man skapar en så kallad *residualgraf* som anger hur stor återstående överföringskapacitet som finns mellan de olika noderna i grafen. Därefter söker man en *utökande väg* genom grafen med kapacitet större än noll och uppdaterar residualgrafen med flödet genom den funna vägen. Detta upprepas tills ingen väg med ledig kapacitet kan hittas.

a) Antag att vi söker det maximala flödet från S till T i en graf med följande utseende och med angivna överföringskapaciteter mellan noderna med hjälp av Ford-Fulkersons metod.



Vi representerar residualgrafen med hjälp av en matris med bågarnas startnoder som rader och slutnoder som kolumner. Hur kommer matrisens innehåll se ut efter de två första stegen i Ford-Fulkersons metod? I det första steget använder vi den utökande vägen S–C–D–T och i det andra steget S–C–E–T. *(3p)*

Svara genom att fylla i matrisen.

	S	A	B	C	D	E	T
S							
A							
B							
C							
D							
E							
T							

---

b) I föregående deluppgift använde vi en matris för att representera residualgrafen. Redogör kortfattat för ett annat sätt vi gått igenom i kursen för att representera en graf i våra datorprogram och som är särskilt lämpligt för grafer där noderna har relativt få grannar. *(1p)*

Skriv ditt svar här.

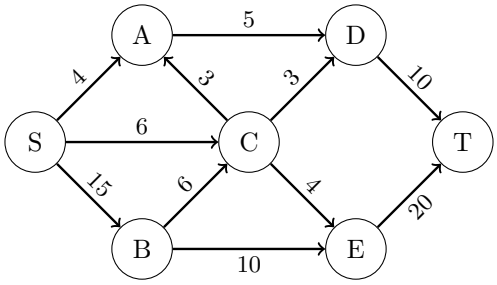
---

c) Enligt Ford-Fulkerson kan man i varje steg välja vilken möjlig utökande väg som helst, men om man ska följa Edmond-Karps beskrivning av metoden ska vägen väljas på ett särskilt sätt. Hur? *(1p)*

Skriv ditt svar här.

Uppgift 2

Låt oss återigen betrakta grafen från föregående uppgift men låt denna gång värdena på bågarna representera *bågarnas vikter (kostnader)*:



a) Antag att vi använder Dijkstras algoritm för att beräkna den lägsta kostnaden att gå från noden S till var och en av de andra noderna. I vilken ordning kommer Dijkstras algoritm att behandla (slutligt bestämma kostnaden för) noderna i grafen? *(2p)*

Skriv ditt svar här.

---

b) I vårt fall har alla bågarna positiva vikter (kostnader), men om någon av vikterna hade varit negativ skulle vi inte kunna använda Dijkstras algoritm. Vad är namnet på den algoritm (som vi nämnt i kursen) vi i så fall hade behövt använda? *(1p)*

Skriv ditt svar här.

Uppgift 3

Roger försöker lösa laborationsuppgift 3 i vilken man bland annat skulle skriva en metod som beräknar och returnerar kortaste avståndet från en nod, *source*, till en annan nod, *dest* (eller -1 om ingen väg finns). Hittills har han lyckats skriva följande kod:

```
1  public static <T> int distance(Graph<T> g, T source, T dest) {
2    Set<T> visited = new HashSet<>();
3    int distance = 0;
4    visited.add(source);
5    Set<T> curLevel = new HashSet<>();
6    curLevel.add(source);
7    while (!curLevel.isEmpty()) {
8      Set<T> nextLevel = new HashSet<>();
9
10     // TODO: I do not know what to write here
11
12   }
13   return -1;
14 }
```

Rogers idé är att använda bredden-först-genomgång och låta `curLevel` representera noderna med det aktuella avståndet från *source* och `nextLevel` noderna som ligger ytterligare ett steg bort.

Hjälp Roger genom att skriva den kod som ska in på kommentarens plats (rad 10)! Skriv svaret på nästa sida. *(2p)*

Interfacet `Graph<T>` ser ut så här:

```
public interface Graph<T> {
  /**
   * Returns the number of vertices in this graph.
   */
  int vertexCount();

  /**
   * Returns the collection of vertices in this graph.
   */
  Collection<T> vertexSet();

  /**
   * Returns the collection of neighbours for vertex v.
   * If v is not part of the graph, an empty list is returned.
   */
  Collection<T> neighbours(T v);
}
```

Skriv ditt svar här.

Fortsätt med dina svar här om du behöver.

Lösningsförslag

Uppgift 1

a)

	S	A	B	C	D	E	T
S	0	4	15	0	0	0	0
A	0	0	0	0	5	0	0
B	0	0	0	6	0	10	0
C	6	3	0	0	0	1	0
D	0	0	0	3	0	0	7
E	0	0	0	3	0	0	17
T	0	0	0	0	3	3	0

b) Man kan använda närhetslistor. Då har man en lista innehållande alla noder. Varje nod har i sin tur en lista bestående av de noder som är grannar till noden samt eventuella vikter för bågarna till grannoderna.

c) Man ska använda bredden-först-sökning för att hitta en så kort väg som möjligt.

Uppgift 2

a) (S,) A, C, D, E, B, T

b) Bellman-Ford

Uppgift 3

```
for (T w:curLevel) {
  if (w.equals(dest)) {
    return distance;
  }
  for (T n:g.neighbours(w)) {
    if (!visited.contains(n)) {
      visited.add(n);
      nextLevel.add(n);
    }
  }
}
distance++;
curLevel = nextLevel;
```