DGA Ü5

1)
```
Algorithm(A, k){
    n:= A.length;   C[0,...,k] neues Datenfeld;
    for i=0,...,k {
        C[i]=0;
    }
    for i=0,...,n{
        C[A[i]] += 1;
    }
    for i=1,...,k{
        C[i] += C[i-1];
    }
}
```

Um nun die Anzahl der Elemente, die in $[a,b]$ liegen zu erhalten wertet man einfach $\underbrace{C[b]}_{\#\{x\in A:\, x\leq b\}} - \underbrace{C[a-1]}_{\#\{x\in A:\, x<a\}}$ aus

$$\#\{x\in A:\, a\leq x\leq b\} = \#\{x\in A:\, x\leq b\} - \#\{x\in A:\, x<a\}$$

(Für $[0,b]$ natürlich $C[b]$.)

DGA Ü5

2) n ganze Zahlen zwischen 1 und n^3-1

ges: Algorithmus, der die Zahlen in $O(n)$ sortiert

Radix-Sort n 3-stellige Zahlen mit jeder Stelle in $\{0,\dots,n\}$

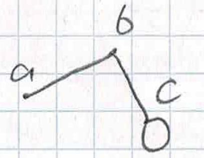
$\Rightarrow$ Laufzeit $\Theta(3(n+n)) = \Theta(6n) = O(n)$

```
Algorithm (A){
  n:= A.length;
  for i=0,...3 {
     B[1,...,n], C[0,...n] neue Datenfelder;
     for j=0,...,n {
        C[j]=0;
     }
     for j=1,...n{
        C[A[j][i]] +=1; // A[j][i] ist die i-te Stelle der j-ten Zahl
     }
     for j=1,...,n{
        C[i] += C[i-1];
     }
     for j=n,...,1 {
        B[C[A[j][i]]] = A[j];
        C[A[j][i]] -= 1;
     }
     copy B to A;
  }
  return A;
}
```

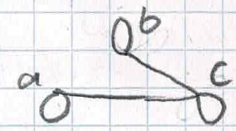
DGA Ü5

3) a) A... Adjazenzmatrix   ges: Bedeutung von A^k

A^k gibt an welche Knoten über k viele Kanten verbunden sind.

Bsp:


$A=\begin{pmatrix}0&1&0\\1&0&1\\0&1&1\end{pmatrix}$ $A^2=\begin{pmatrix}1&0&1\\0&2&1\\1&1&2\end{pmatrix}$



Die 2 bedeutet, es gibt 2 unterschiedliche Verbindungen der Kanten (z.B. $b \to a \to b \wedge b \to c \to b$).

Beweis: Vollständige Induktion nach k:

$k=1$: klar

$k+1$: $A^k[i,j] = \sum_{l=0}^{n} A^{k-1}[i,l] \cdot A[l,j]$

Nach Induktionsvorraussetzung gibt $A^{k-1}[i,l]$ an wie viele verschiedene $k-1$-lange Verbindungen es zwischen i und l gibt. Falls $A[l,j]=1$ gibt es also eine k-lange Verbindung zwischen i und j. Falls $A[l,j]=0$ gibt es keine.

$\Rightarrow \sum_{l=0}^{n} A^{k-1}[i,l] \cdot A[l,j]$ summiert die Anzahl an k-langen Verbindungen zwischen i und j

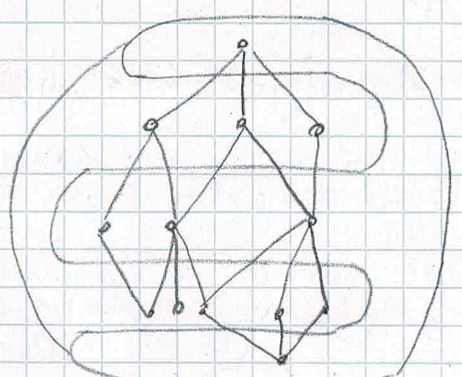
b) G... ungerichtete Graph, ohne Schlingen und Mehrfachkanten

A... Adjazenzmatrix von G

ges: Anzahl Zyklen der Länge 3

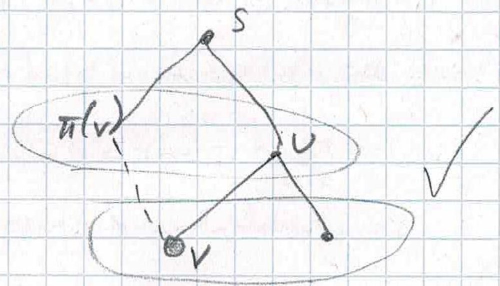
$\sum_{l=0}^{n} A^3[l,l]$, da A^3 angibt wie viele Kantenfolgen der Länge 3 zw. je zwei Knoten existieren. Damit es Zyklen der Länge 3 sind muss der Anfangsknoten = Endknoten sein, also interessiert uns nur die Diagonale. Um die insgesamte Anzahl zu erhalten summieren wir über die Diagonale von A^3.

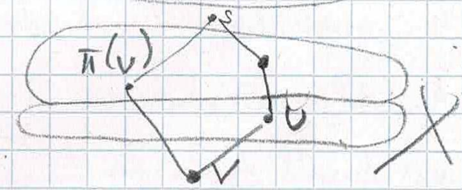
DGA Ü5



4)

```
Algorithmus(G, s) {
  f(s)=GRAU; d(s)=0; π(s)=NIL;
  V1.append(s);
  ENQUEUE(Q,s);
  for u∈V\{s} {
    f(u)=WEISS; d(u)=∞; π(u)=NIL;
  }
  while Q ≠ [] {
    u=DEQUEUE(Q)
    for v∈Adj(u) {
      if f(v)==WEISS {
        f(v) = GRAU; d(u)=d(u)+1; π(v)=u;
        if d(v)%2 == 0 {
          V1.append(v);
        }else{
          V2.append(v);
        }
        ENQUEUE(Q,v);
      }else{
        if(d(π(v))%2 != d(u)){return "nicht bipartit"}
      }
    }
    f(u)=SCHWARZ;
}
return(V1, V2);
```



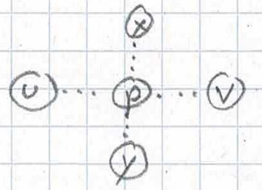


DGA Ü5

5)
```
DFS (G, s, d=0) {
    c(s) = SCHWARZ;
    if (d > max_d) {
        max_node = s; max_d = d;
    }
    for v ∈ Adj(s) {
        if c(v) = WEISS {
            DFS(G, v, d+1);
        }
    }
}

find_diameter (G, s) {
    max_d = -1; max_node = NULL;
    for u ∈ V { c(u) = WEISS; }
    DFS(G, s);
    max_d = -1;
    for u ∈ V { c(u) = WEISS; }
    DFS(G, max_node);
    return max_d;
}
```

Sei x die max_node nach dem ersten DFS und y die max_node nach dem zweiten DFS.
Angenommen der Durchmesser ist u-v mit $u \neq x \neq v \wedge u \neq y \neq v$.



Sei ab die Länge zwischen a und b.
Da y am weitesten entfernt von x ist

$\Rightarrow xp + py > xp + pu \Rightarrow py > pu$

$xp + py > xp + pv \Rightarrow py > pv$

Da der Durchmesser u-v ist

$\Rightarrow up + pv > up + px \Rightarrow pv > px$

$up + pv > up + py \Rightarrow pv > py$

Widerspruch ($py > pv \wedge pv > py$)

$\Rightarrow$ x-y ist der Durchmesser

Aufwand DFS: $O(|V| + |E|)$

Aufwand find_diameter: $O(|V|) + O(|V| + |E|) + O(|V|) + O(|V| + |E|) = O(4|V| + 2|E|) = O(|V| + |E|)$