Werkcollege 5
Brute force, exhaustive search en backtracking

Jan van Rijn

Leiden Institute of Advanced Computer Science

10 maart 2011

Opgave 1

Levitin 3.2.9

Beschouw het probleem waarin je, gegeven een tekst, het aantal substrings moet tellen die met een A beginnen en op een B eindigen. (Bijvoorbeeld CABAAXBYA heeft er hier 4 van.)

- Ontwerp een brute-force algoritme voor dit probleem en bepaal de efficiëntie hiervan.
- Ontwerp een efficienter algoritme voor dit probleem.

Opgave 1A

```
int teller = 0;
for ( int i=0; i<n-1; i++ ) {
  if ( text[i] == 'A' ) {
    for ( int j=i+1; j<n; j++ ) {
      if ( text[j] == 'B' ) {
        teller++;
      } // if B
    } // for j
  } // if A
} // for i
// teller geeft nu het gevraagde aantal substrings
```

Opgave 1B

```
int totaalteller = 0;
int Ateller = 0;
for ( int i=0; i<n; i++ ) {
  if ( text[i] == 'A' )
    Ateller++;
  if ( text[i] == 'B' )
    totaalteller+=Ateller;
} // for i
// totaalteller geeft nu het gevraagde aantal
// substrings
```

Opgave 2

Levitin 3.4.6

Beschouw het partitie probleem. Gegeven n positive integers, verdeel ze in twee niet overlappende groepen met dezelfde som van waardes. (Merk op dat het probleem niet altijd een oplossing heeft.) Ontwerp een exhaustive search algoritme voor dit probleem. Probeer het aantal subsets dat moet worden bekeken, zo klein mogelijk te houden.

Opgave 3

Levitin 3.4.9

Een magisch vierkant van orde n is een vierkant van grootte n bij n waarin de getallen 1 tot en met n dusdanig zijn gerangschikt, dat de kolommen, de rijen en de beide diagonalen allen dezelfde som opleveren.

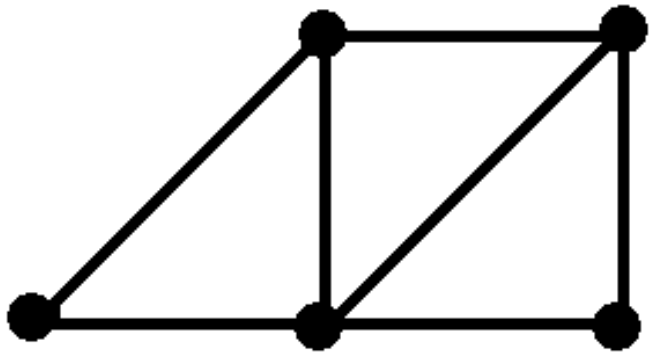
- Bewijs dat wanneer er een magisch vierkant van orde n bestaat, de betreffende som gelijk is aan

$$\frac{n(n^2+1)}{2}$$

.

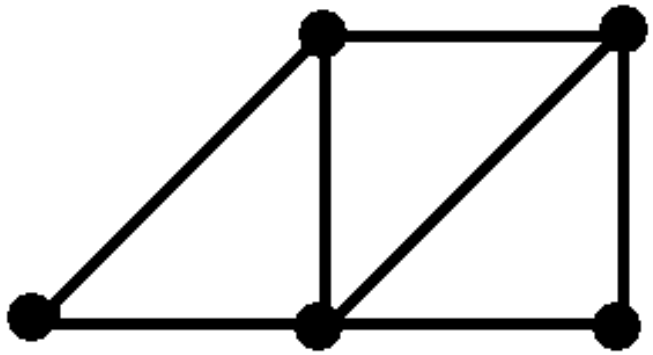
- Ontwerp een exhaustive search algoritme dat alle magische vierkanten van orde n genereert.

Opgave 6

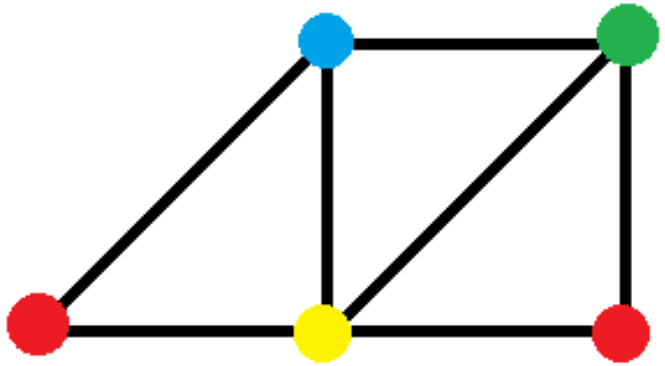


(a)

Opgave 6



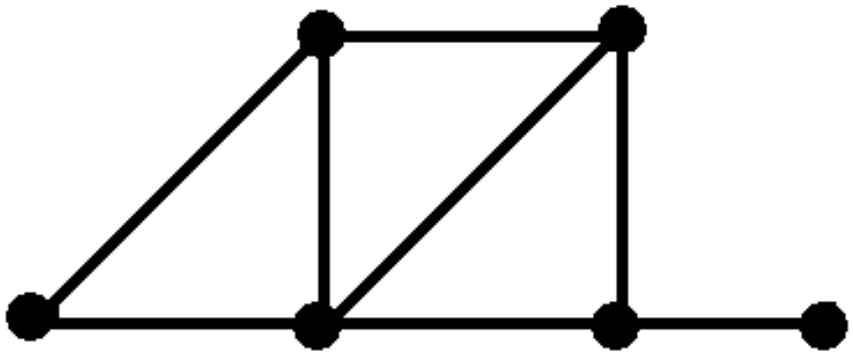
(a)



(b)

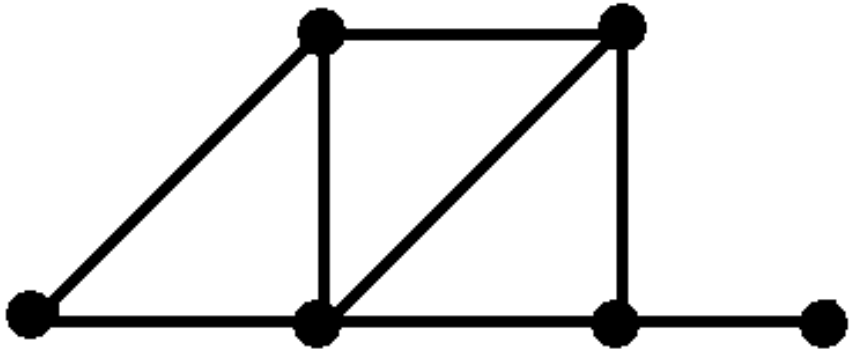
Figure:

Opgave 6

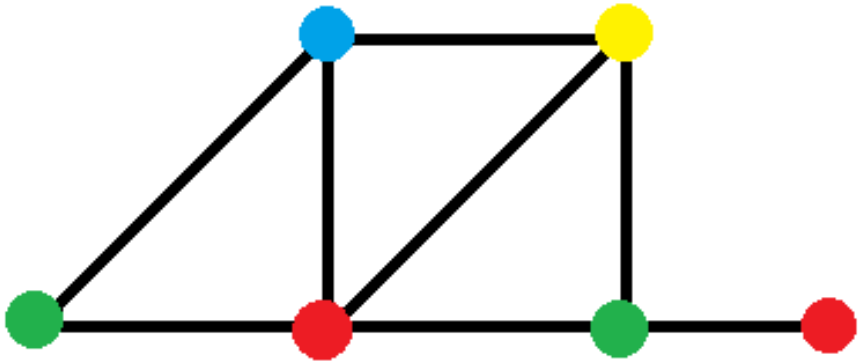


(a)

Opgave 6



(c)



(d)

Opgave 7

```
void latijnsvierkant (int n, int A[n][n], int i, int j) {
   int getal, k;
   bool okee;

   if ( i == n )
       drukaf (n, A);
       // i.pv. afdrukken zou je hier ook het aantal
       // Latijnse vierkanten kunnen tellen
   else {
       for ( getal = 1; getal <= n; getal++ ) {
       // controleer of getal al in rij i of kolom j staat
           okee = true;
           for ( k = 0; k < j; k++ ) {
               if ( A[i][k] == getal )
                  okee = false;
           } // for
           if ( okee )
               for ( k = 0; k < i; k++ ) {
                   if ( A[k][j] == getal )
                       okee = false;
               } // for
           if ( okee ) {
               A[i][j] = getal;
               if ( j < n-1 )
                   latijnsvierkant (n, A, i, j+1);
               else
                   latijnsvierkant (n, A, i+1, 1);
           } // if
        } // for
    } // else
} // einde
```

Opgave 8

```
void partitie (int part[ ], int gedaan, int aantaldelen, int getal,
               int vanaf) {
// Maakt alle partities van getal in aantaldelen, elk deel >= vanaf.
// Al gedaan: gedaan delen. Resultaat komt steeds in part.

  int j;
  if ( ( getal == 0 ) && ( aantaldelen == 0 ) )
     drukaf (part,gedaan); // nieuwe partitie gevonden
  else
     if ( ( getal != 0 ) && ( aantaldelen != 0 ) )
       for ( j = vanaf; j <= getal; j++ ) {
         // of j <= getal - (aantaldelen-1) * vanaf
          part[gedaan+1] = j;    // volgend deel wordt j
          partitie(part, gedaan+1, aantaldelen-1, getal-j, j)
       } // for
}  // partitie
```