10.ud/00.50

1(a)

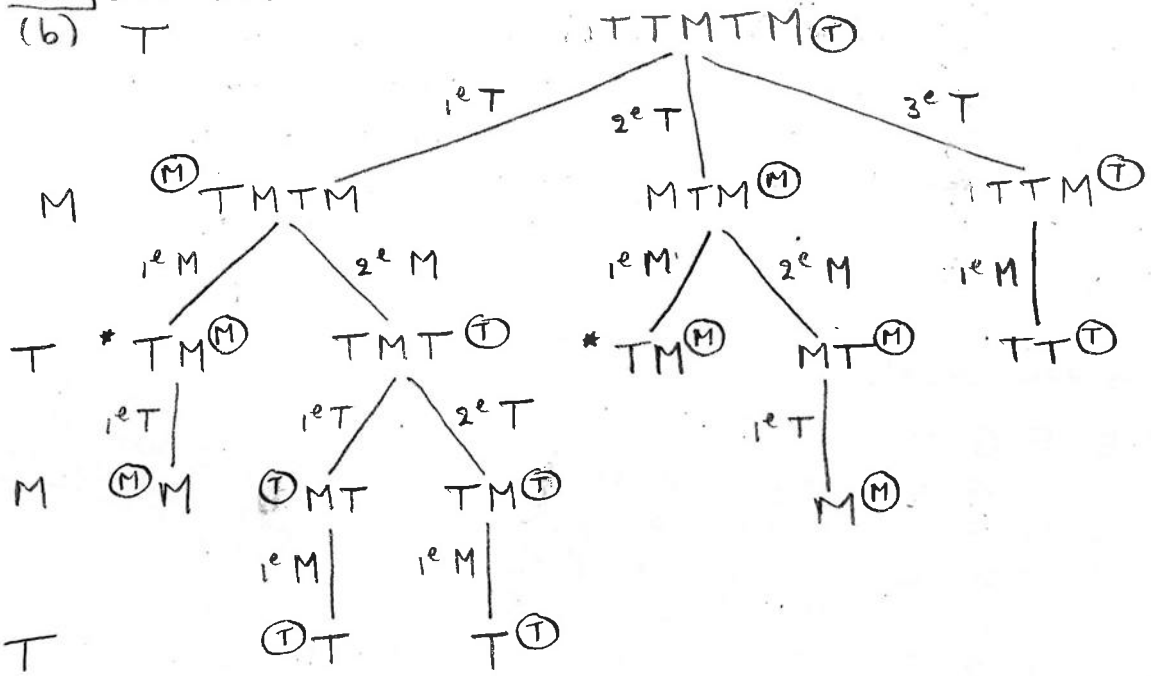
Bij een gegeven beginrij met stenen bestaan de toestanden uit
* een aaneengesloten deelrij van de beginrij met stenen
* een speler die aan de beurt is: Tim of Maarten.

De acties bestaan uit
* het kiezen van een steen van de speler die aan de beurt is
* het verwijderen van de gekozen steen, en het verwijderen van de kortste deelrij met stenen of (als beide deelrijen even lang zijn) het verwijderen van de rechter deelrij met stenen
* het wisselen van beurt.

Een toestand is een eindtoestand als er in die toestand alleen nog stenen van Tim of stenen van Maarten over zijn

1.00 aan beurt

(b)



* = al eerder uitgewerkt / zelfde toestand
(T) = winnend voor Tim
(M) = winnend voor Maarten

Het spel is dus winnend voor Tim

11.11/01.15/01.20

Merk op: Tim is als eerste aan de beurt

(c) Als $m \geq n$, is het spel winnend voor Maarten
Als $m < n$, is het spel winnend voor Tim

Een zo goed mogelijke zet voor de speler die aan de beurt is, bestaat uit de keuze (en het verwijderen) van zijn buitenste steen. Daarmee verliest hij maar 1 steen, terwijl het aantal stenen van de tegenstander gelijk blijft.

Als je een steen kiest die meer naar binnen in de rij ligt, zul je doorgaans meer stenen kwijtraken (niet alleen de gekozen steen, maar ook de stenen 'aan de buitenkant van' de gekozen steen), terwijl de tegenstander nog steeds al zijn stenen behoudt. Uitzondering hierop is, wanneer je minstens twee stenen meer hebt dan de tegenstander, of in het geval van Maarten ook als hij één steen meer heeft dan Tim. Als je dan je binnenste steen kiest, de steen die tegen de stenen van de tegenstander aanligt, zul je in één klap alle stenen van de tegenstander elimineren, en dus direct winnen. Echter, in deze situatie zou je nog steeds winnen, als je kiest voor je buitenste steen, alleen niet in één klap.

)1.36

4(a) i) Even snel in een tabel

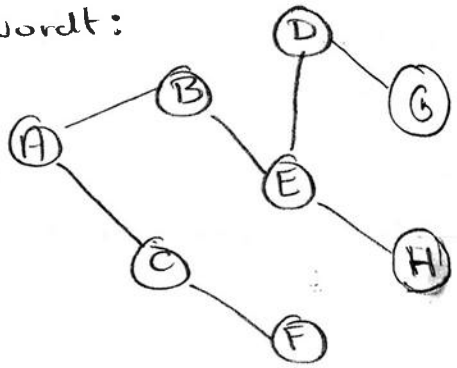
A	B	C	D	E	F	G	H	
0	∞	∞	∞	∞	∞	∞	∞	begin met A
–	4	3	∞	∞	∞	∞	∞	kies C, vanaf A
–	4	–	∞	9	8	∞	∞	kies B, vanaf A
–	–	–	10	5	8	∞	∞	kies E, vanaf B
–	–	–	9	–	8	12	9	kies F, vanaf C (niet vanaf E ?)
–	–	–	9	–	–	12	9	kies D, vanaf E (of H vanaf E)
–	–	–	–	–	–	11	9	kies H, vanaf E
–	–	–	–	–	–	11	–	kies G, vanaf D

Want E leverde geen verbetering

)1.52 Volgordes 2 en 3 zijn goed. De andere twee niet

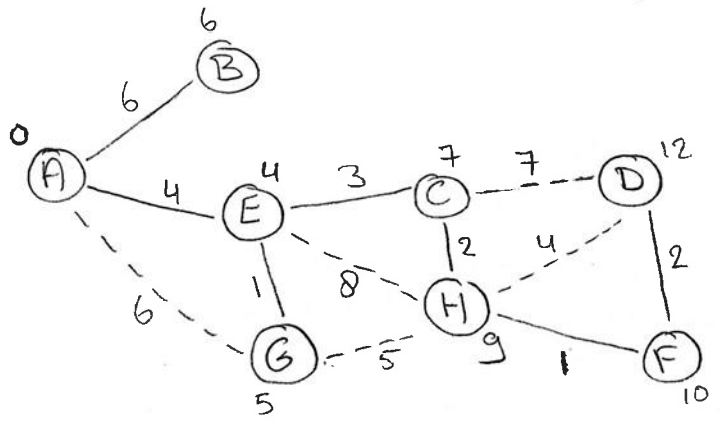
ii) De takken in de boom van kortste paden liggen vast, bij dit voorbeeld ook al kan de volgorde van de knopen variëren.

Het wordt: Dat is boom 3

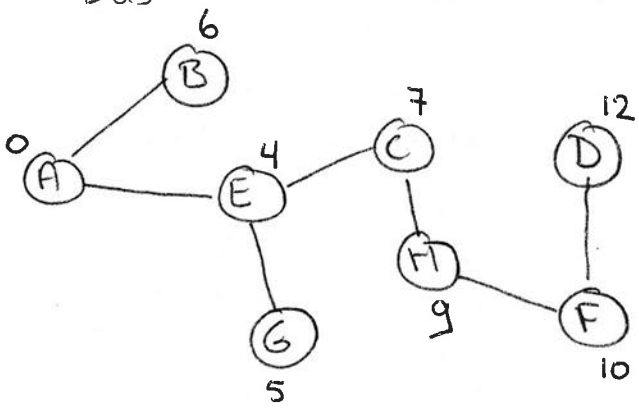


1.57

(b)



02.05.
Dus



02.06

09.24

2 (a)

```
int piekdallen1 (int A[], int N)
{    // pre: N >= 2
int teller = 2;   // A[0] en A[N-1]
   for (int i=1; i<N-1; i++)
   { bool stijg1 = !(A[i-1] < A[i]); bool stijg2 = (A[i] < A[i+1]);
     if (stijg1 != stijg2)  // A[i] is een piek-dal
       teller ++;
   } // for
   return teller;
}
```

(b)

```
int piekdallen2 (int A[], int i)
{    // pre: i >= 2
   int teller;
   if (i == 2)        // A[0] en A[1] zijn piek-dallen van het deelarray.
     return 2;
   else   // i > 2
   { teller = piekdallen2(A, i-1);
     // bij recursieve aanroep is A[i-2] als piek-dal geteld van het
     // deelarray; controleer of hij dat ook in het grotere array is.
     bool stijg1 = (A[i-3] < A[i-2])
     bool stijg2 = (A[i-2] < A[i-1])
     if (stijg1 == stijg2)  // A[i-2] is toch geen piek-dal
       teller --;
     teller ++;  // A[i-1] is sowieso piek-dal in dit grotere deelarray
     return teller;
   }
}
```

(c)
```
int piekdallen3 (int A[ ], int links, int rechts)
{  // pre: links < rechts en rechts-links+1 is een twee-macht
   int teller, mid;
   bool stijg1, stijg2, stijg3;

   if (links+1 == rechts)  // nog twee elementen over
     return 2;    // A[links] en A[rechts] zijn piek-dallen in het deelarray
   else
   { mid = (links + rechts) / 2;   // links van het midden
     teller = piekdallen3 (A, links, mid) + piekdallen3 (A, mid+1, rechts);
     // bij recursieve aanroepen zijn A[mid] en A[mid+1] als piek-dallen
     // geteld van de deelarrays; controleer of ze dat ook in het grotere
     // array zijn
     stijg1 =  (A[mid-1] < A[mid]);
     stijg2 =  (A[mid] < A[mid+1]);
     stijg3 =  (A[mid+1] < A[mid+2]);
     if (stijg1 == stijg2)    // A[mid] is geen piek-dal
       teller --;
     if (stijg2 == stijg3)    // A[mid+1] is geen piek-dal
       teller --;
     return teller;
   } // else
}
```

09.47

11.24 / 11.26

3 (a) Gretig algoritme: maak een getal rood, tenzij het niet mag

rood: 3, 4, 6, 8, 12, 13

blauw: 5, 10

De drie Pythagorese drietallen met getallen in A zijn:
(3, 4, 5), (6, 8, 10) en (5, 12, 13).

11.30

b)

```
bool uitbreidingOK (int A[], bool rood[], int k)
{ int i, j;

  i=0;
  j=k-1;
  while (i<j)
  {
    if (square(A[i]) + square(A[j]) == square(A[k])
        && rood[i] == rood[k] && rood[j] == rood[k])
      return false;
    if (square(A[i]) + square(A[j]) < square(A[k])
      i++;   // het heeft niet meteen zin om j te verlagen,
    else     // want dan wordt de som van de kwadraten
      j--;   // nog kleiner
             // analoog voor i verhogen
  }

  // geen probleem ontdekt
  return true;
}
```

1.43

(c)

```
int aantalkleuringen (int A[], int N, bool rood[], int k)
{ int teller;
  if (k == N)   // kleuring compleet
    return 1;
  else   // k<N
  { teller = 0;
    rood[k] = true;
    if (uitbreidingOK (A, rood, k))
      teller += aantalkleuringen(A, N, rood, k+1);
    rood[k] = false;
    if (uitbreidingOK (A, rood, k))
      teller += aantalkleuringen (A, N, rood, k+1);
    return teller;
  }
}
```

11.50

10.41

(d)

Als we een geschikte kleuring hebben waarbij A[0] rood is, kunnen we daar een andere geschikte kleuring van maken door alle rode getallen blauw te maken en omgekeerd. In die nieuwe kleuring is in het bijzonder A[0] blauw.

Net zo kunnen we van elke geschikte kleuring waarin A[0] blauw is een geschikte kleuring maken waarin A[0] rood is.

Er is dus een 1-1 correspondentie tussen geschikte kleuringen waarin A[0] rood is en geschikte kleuringen waarin A[0] blauw is. Oftwel, er zijn er evenveel.

Bij het berekenen van het totaal aantal geschikte kleuringen kunnen we ons daarom beperken tot kleuringen waarin A[0] rood is, en vervolgens het resultaat met 2 vermenigvuldigen.

10.48