

Oscar Azrak  
oazrak@kth.se

DD1320 Tillämpad datalogi - KTH HT 2020

# 1 Uppgiftsbeskrivning

Matchsticks are ideal tools to represent numbers. A common way to represent the ten decimal digits with matchsticks is the following:

1 2 3 4 5 6 7 8 9 0

This is identical to how numbers are displayed on an ordinary alarm clock. With a given number of matchsticks you can generate a wide range of numbers. We are wondering what the smallest and largest numbers are that can be created by using all your matchsticks.

## Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with an integer  $n$  ( $2 \leq n \leq 100$ ): the number of matchsticks you have.

## Output

Per testcase:

- One line with the smallest and largest numbers you can create, separated by a single space. Both numbers should be positive and contain no leading zeroes.

## 2 Kattis

<https://kth.kattis.com/submissions/6507271>

## 3 Datastrukturer

De datastrukturer som använts i detta program är listor och integers (heltal).

## 4 Algoritm

Algoritmen sparar antalet tändstickor en siffra kräver i lista från noll till nio vilket gör att indexet motsvarar vilken siffra som motsvarar antalet tändstickor. Sedan bestäms det största och minsta värdet i listan vilket motsvarar alltså mest antal tändstickor som krävs för en siffra och minsta antalet tändstickor som krävs för en siffra. Sedan skapas två funktioner, en som beräknar den minsta talet och en som beräknar det största talet. För min-funktionen: I bägge funktionerna bestäms hur många siffror som krävs, i min-funktionen beräknas antalet siffror med  $\text{math.ceil}(\text{numStickor}/7)$  där numStickor är antalet tändstickor som ska undersökas. Detta betyder att modulen math måste först importeras i algoritmen så att man kan använda math.ceil(). Math.ceil avrundar ett tal uppåt, vilket används i det här fallet antal tändstickor dividerat med maxStickorEnSuffra för att få hur många siffror som kan användas. Från 2-7 går det bra med en siffra, 8-14 2 går siffror och så vidare. Sedan sparas en variabel för hur många tändstickor som är kvar, i början bör denna motsvara lika mycket som antalet stickor som ska undersökas. Sedan undersöks hur många stickor som är lämpligast för varje siffra för att få så litet tal som möjligt, så den viktigaste siffran undersöks först till den minst viktigaste siffran vilket är den första till sista siffran. Algoritmen tar fram både minsta antalet stickor man kan använda och max antal stickor man kan använda för respektive siffra.

För att få minsta antalet stickor används en inbyggd max-funktion som jämför (minsta antal stickor en siffra kräver) med (stickor som är kvar minus max stickor en siffra kräver gånger antalet siffror minus 1 minus siffran) där siffran är från noll till (antalSiffror - 1) och returnerar det största talet av dessa två

tal, om de är lika returneras den som dtår först. För att få max stickor en siffra kan använda används en inbyggd min-funktion som jämför (max stickor en siffra kan använda) med (stickor kvar minus min stickor en siffra kan använda gånger antal siffror – 1 minus siffran) och returnerar den minsta talet av dessa två tal.

Sedan bestäms den minsta talet som kan användas för just den siffran, eftersom man ej får ha ledande nollor måste man separera två fall, ett fall då resultatet är noll, vilket definieras från början av funktionen och ett fall då det ej är noll. I det första fallet exkluderas noll från att hittas i listan. För att hitta det passande talet till letar algoritmen efter ett tal i listan som är större eller lika med minsta antal stickor man kan använda och mindre eller lika med max antal stickor man kan använda. När denna siffra hittas så adderas denna siffra till resultatet genom att addera indexet till det talet. Talet adderas för varje siffra genom att multiplicera tidigare resultat med 10 och addera det index som listan hamnar på. Alternativt kan man addera de som strings, då behöver man ej multiplicera med 10. Sedan subtraheras det talet från antalet stickor som är kvar.

Algoritmen upprepar denna process tills alla siffror fått sitt tal.

Max-funktionen och min-funktionen är väldigt lika men exempel på skillnader är för att få antalet siffror, i max-funktionen vill man ej avrunda uppåt utan man avrundar nedåt vilket betyder att man kan dividera till hel tal med hjälp av att använda dubbel dividering i python. Eftersom man vill använda så många siffror med så få tändstickor som möjligt divideras antalet stickor med minStickorEnSiffra, detta gör att exempelvis 4 stickor använder två siffror.

Processen för att hitta minsta respektive max antal stickor att använda per siffra är identisk från min-funktionen.

Processen att hitta lämpligt tal för varje siffra skiljer sig i max-funktionen från min-funktionen. Man kan se som ett mönster att den första siffran i talet är antingen 7 eller 1 vilket betyder att när resultat är lika med noll så undersöks samma lista men istället för att gå igenom listan från index noll till nio undersöks listan baklänges alltså från nio till noll för att hitta det största talet först och inte sist. Om man ej ändrar på detta kommer sjuan skrivas ut sist i talet istället för först vilket gör att man missar den verkliga största siffran. Sedan indexet med resultatet på identiskt sätt som min-funktionen. Sist subtraheras stickorKvar med det antal stickor som använts.

Sist skapades en main-funktion som knöt ihop dessa två funktioner. I main-funktionen ber den först användaren om en input, som ska vara hur många rader kommer efter den raden så att programmet vet hur många gånger algoritmen ska köras. En for loop körs och där sparas användarens input i en tom lista. Sedan loopas listan igenom igen i en separat for loop där min-och max-funktionerna anropas och skriver ut svaren.

## 5 Tidskomplexitet

I min kod har jag en två funktioner med två stycken for-loops i varandra. Loopar från 0 till n medan den andra loopar från 1 till 10 eller 0 till 9. Detta gör att tidskomplexiteten blir  $O(10*n)$  där O, "Big O Notation" används för att klassificera algoritmer enligt hur snabba de är eller hur mycket utrymme som krävs och n är storleken. I det här fallet används O-noteringen för att avgöra tidskomplexiteten. I "main" funktionen finns det dessutom en till for-loop som loopar från 0 till n, beroende på hur många inputs man ska ta emot. Denna funktion har tidskomplexitet  $O(n)$ , detta gör att man kan addera  $O(10*n)$  med  $O(n)$  som blir  $O(11*n)$  men detta kan förenklas till  $O(n)$  eftersom n kan bli hur stort som helst så spelar konstanterna ingen roll.

Tidskomplexiteten för min kod är  $O(n)$

## Appendix 1: Programkod

```
1 import math
2
3 def hashfunktion(namn):
4     summa = 0
5     for tkn in namn:
6         summa = summa * 365 + ord(tkn)
7     return summa % 7 + 1
8 # print(hashfunktion("qazrak"))
9
10 # Uppgift: Matchsticks
11
12
13 #funktion ger minstavärde med antal stickor
14 def minSiffra(numStickor, maxStickorEnSiffra, minStickorEnSiffra, stickLista):
15     antalSiffror = math.ceil(numStickor/7) #Avrundar värdet upp för att få hur många siffror som krävs
16     stickorKvar = numStickor
17     resultmin = 0
18
19     for i in range(0, antalSiffror):
20         #max-funktion används för att jämföra vilken siffra som e störst mellan minStickorEnSiffra eller antalet stickor
21         #som är kvar minus maxstickor gånger antalsiffror-1, x = 7(x-1) sedan subtraheras även i för varje gång
22         minStickorAttAnvanda = (max(minStickorEnSiffra, stickorKvar - maxStickorEnSiffra * (antalSiffror - 1 - i)))
23
24         #liknande till minStickorAttAnvanda sker exakt samma men för maxStickorAttAnvanda,
25         #nu används en min funktion istället och minStickorEnSiffra, 2
26         maxStickorAttAnvanda = (min(maxStickorEnSiffra, stickorKvar - minStickorEnSiffra * (antalSiffror - 1 - i)))
27
28         #Första siffran får ej börja med noll så en if-sats för att hindra resultatet från att aldrig hitta noll,
29         #annars skrivs ex 168 ut istället för 108 för 15 stickor,
30         if resultmin == 0:
31             for index in range(1, 10):
32                 if minStickorAttAnvanda <= stickLista[index] <= maxStickorAttAnvanda:
33                     #resultatet mult. med 10 för att kunna addera rätt, alternativt hade det funkat att addera som stri
34                     resultmin = resultmin * 10 + index
35                     #Antal stickor som är kvar subtraheras med den motsvarande siffran till index,
36                     #så om index=0 subtraheras 6
37                     stickorKvar -= stickLista[index]
38                     break
39             else:
40                 for index in range(0, 10):
41                     if minStickorAttAnvanda <= stickLista[index] <= maxStickorAttAnvanda:
42                         resultmin = resultmin * 10 + index
43                         stickorKvar -= stickLista[index]
44                         break
45         return resultmin
46
47
```

```

48 #Returnerar högsta siffran med numStickor st tändstickor
49 def maxSiffra(numStickor, maxStickorEnSiffra, minStickorEnSiffra, stickLista):
50     #antal stickor dividerat med minsta antal stickor för en siffra, avrundas nedåt, math.floor hade kunnat användas här
51     #för att få hur många siffror som ska användas
52     antalSiffror = numStickor // minStickorEnSiffra
53
54     stickorKvar = numStickor
55     resultmax = 0
56     #for-loop antalSiffror gånger för att hitta det största värde per siffra som sedan läggs ihop som resultatmax
57     for i in range(0, antalSiffror):
58         minStickorAttAnvanda = (max(minStickorEnSiffra, stickorKvar - maxStickorEnSiffra * (antalSiffror - 1 - i)))
59         maxStickorAttAnvanda = (min(maxStickorEnSiffra, stickorKvar - minStickorEnSiffra * (antalSiffror - 1 - i)))
60         if resultmax == 0:
61             #kollar sticklistan från slutet till början för att nu bör den högsta siffran vara i början och ej i slutet
62             for index in range(9, 0, -1):
63                 if minStickorAttAnvanda <= stickLista[index] <= maxStickorAttAnvanda:
64                     resultmax = resultmax * 10 + index
65                     stickorKvar -= stickLista[index]
66                     break
67             else:
68                 for index in range(0, 10):
69                     if minStickorAttAnvanda <= stickLista[index] <= maxStickorAttAnvanda:
70                         resultmax = resultmax * 10 + index
71                         stickorKvar -= stickLista[index]
72                         break
73         return resultmax
74
75 #kollar om antal rader n input är korrekt
76 def checknFOTOK(n): #function that checks wether the size of the board is correct
77     nOK = False
78     try:
79         n = int(n)
80         if n > 0:
81             nOK = True
82         else:
83             pass
84     except ValueError:
85         pass
86     return nOK
87
88 #undersöker om inputs är korrekt
89 def checkNumStickorOK(numStickor):
90     numStickorOK = False
91     try:
92         numStickor = (numStickor)
93         if 2 <= int(numStickor) <= 100:
94             numStickorOK = True
95         else:
96             pass
97     except ValueError:
98         pass
99     return numStickorOK
100
101
102

```

```

103 #tar emot input från användaren och undersöker om det är korrekt
104 def numSticks():
105     numStickorOK = False
106     while numStickorOK == False:
107         numStickor = input()
108         numStickorOK = checkNumStickorOK(numStickor)
109     return numStickor
110
111 def main():
112     # antal stickor som behövs från 0-9, första siffran är hur många stickor siffran 0 behöver
113
114
115     #lista med antalet stickor per index, index 0 kräver 6 stickor, osv
116     stickLista = [6, 2, 5, 5, 4, 5, 6, 3, 7, 6]
117
118     # största talet i listan
119     maxStickorEnSiffra = max(stickLista)
120     # minsta talet i listan
121     minStickorEnSiffra = min(stickLista)
122
123     #kollar om antal rader man ger i input är rätt, måste vara över noll och en siffra
124     nOK = False
125     while nOK == False:
126         n = input()
127         nOK = checknFTOK(n)
128
129     #korlistan sparar vilka indata som ska undersökas
130     korListan = []
131
132     #for-loop som går igenom inputet n antal gånger från första input värdet, kollar även med hjälp av
133     #numSticks() funktionen om input är korrekt
134
135     #loopar igenom n gånger för att få inputsen, kollar även om inputsen är rätt, lägger till inputsen i korlistan
136     for i in range(0,int(n)):
137         numStickor = numSticks()
138         korListan.append(int(numStickor))
139
140     #for-loop som printar resultat för varje input, minsta värde respektive största värdet.
141     for siffran in korListan:
142         print(minSiffra(siffran, maxStickorEnSiffra, minStickorEnSiffra, stickLista),
143               maxSiffra(siffran, maxStickorEnSiffra, minStickorEnSiffra, stickLista))
144
145     main()
146

```

## Appendix2: Testdata

Den första testdata som testades är om första input är noll eller mindre ska programmet be om en ny input tills en godkänd input ges då ska programmet ta emot input för antalet stickor.

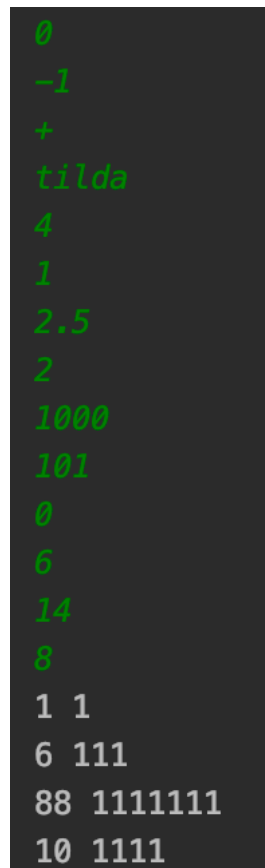
Den andra testdatan är om man anger noll stickor eller en sticka, då ska programmet be om ny input tills användaren ger korrekt input.

Och den tredje testdatan är att testa om programmet tillåter ledande nollor, så först testas 6 stickor och sedan 14 och 8, om den skulle tillåta med ledande nollor är den lägsta siffran 001 respektive 01.

I alla tre fall får endast inputs som kan omvandlas till integers tillåtas, inga bokstäver, specialtal eller speciella tecken.

Så testdatan är:

0            ← testar om antal rader som ska läsas in efter är 1 eller mer, om de är mindre provas en input  
-1           ← testar som ovan, om siffran man ger i input är mindre än 1 ska den försöka igen  
+            ← programmet ska endast tillåta siffror  
tilda        ← testar som ovan, programmet ska endast tillåta siffror  
4            ← en tillåten input  
1            ← antal tändstickor måste vara två till hundra, så under 2 tillåts ej  
2.5          ← Tillåter ej decimaltal  
2            ← tillåten input  
1000        ← som ovan, input måste vara mellan 2 och 100  
101  
0            ← testar att programmet ej tillåter fel input trots en tillåten input angetts tidigare  
6            ← får ej ge 0 som svar, måste ge 6  
14           ← testar om den ger 001 eller 88 som den bör  
8            ← testar om den ger 01 eller 10 som den bör



```
0
-1
+
tilda
4
1
2.5
2
1000
101
0
6
14
8
1 1
6 111
88 1111111
10 1111
```

Programmet ignorerar de felaktiga inputsen och tar endast in de korrekta inputs och ger do ut korrekta outputs