PROGRAMMERINGS OLYMPIADEN

PO-Finalen 2022

Lösningsförslag

Badstrand

Av: Fredrik Ekholm och Abdullah Zaghmout

Testfallsgrupp 1 (N ≤ 1000 och alla A_i har samma värde)

Här kostar alla tomter samma pris. Det spelar därför ingen roll var vi börjar. Vi kan börja från första tomten och köpa tomter tills vi inte längre har tillräckligt med pengar för att köpa en ny tomt, eller tills vi har köpt alla tomter. Tidskomplexiteten: O(N)

En snabbare lösning är att använda sig av matte. Antalet tomter man maximalt kan köpa om en tomt kostar A och vårt budget är B kronor är ⌊B/A⌋. Svaret blir då min(⌊B/A⌋,N). Tidskomplexiteten: O(1)

Testfallsgrupp 2 (N ≤ 1000)

Här spelar det roll var vi faktiskt börjar. Men eftersom N är liten så kan vi prova alla platser att börja på. När vi börjar vid en viss plats så köper vi alla tomter tills vi inte längre har tillräckligt med pengar för att köpa nästa tomt eller tills vi har kommit till slutet av kusten.
Tidskomplexiteten: O(N^2)

Testfallsgrupp 3 (N ≤ 100 000)

Här kan vi inte prova alla platser att börja på, utan vi måste göra något smartare. Vi använder en teknik kallad för två pekare teknik.

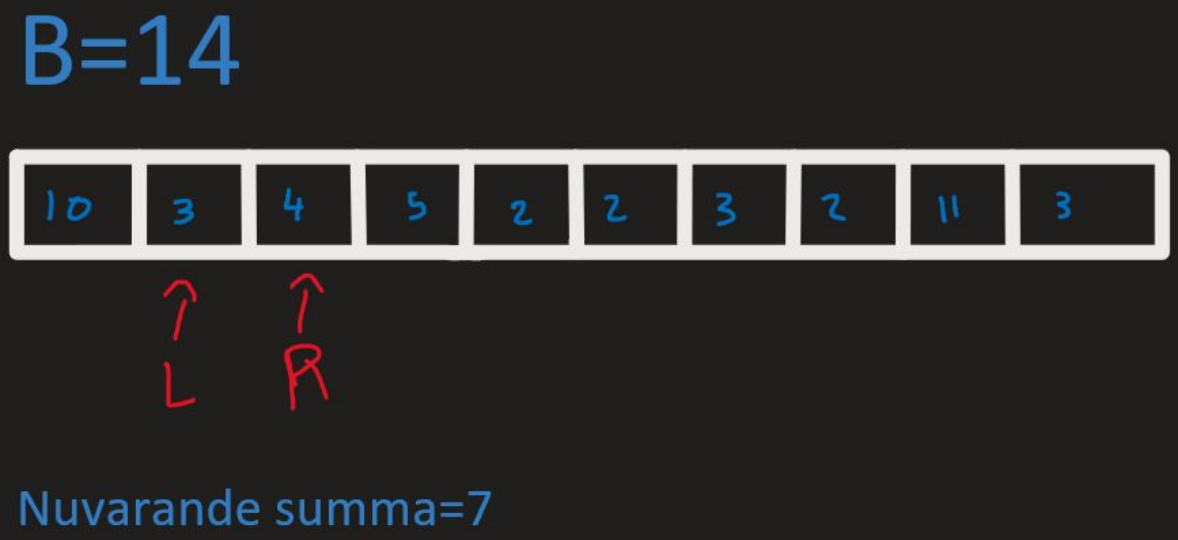
Exempel:

Testfallsgrupp 3 (N ≤ 100 000)

Nästa iteration:

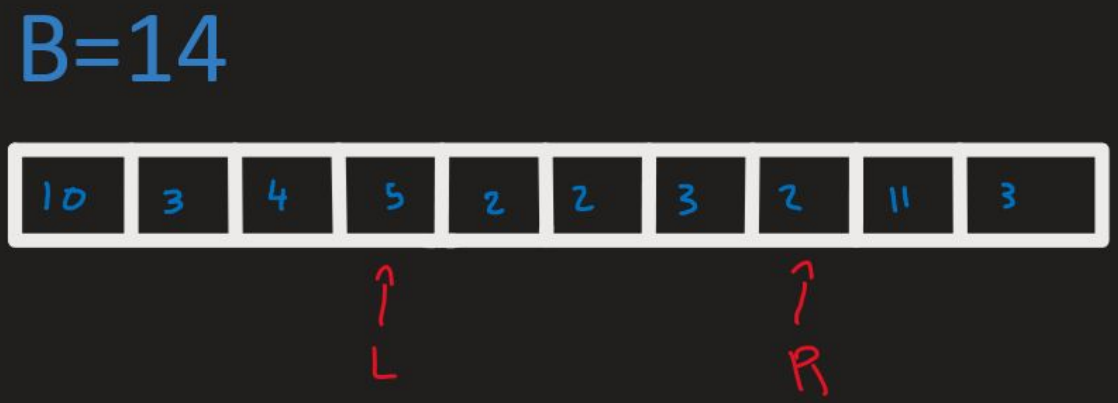
Testfallsgrupp 3 (N ≤ 100 000)

När vi inte längre kan köpa nästa tomt minskar vi nuvarande summa med A[L] och vi okär L(vänster pekaren) med 1.



Testfallsgrupp 3 (N ≤ 100 000)

Iterationen som ger oss svaret:



Kortlek

Av: Abdullah Zaghmout

Testfallsgrupp 1 (N ≤8)

Här kan vi se att N är jätteliten och därmed så är M jätteliten också. Vi provar alla permutationer och väljer det bästa resultatet. Tidskomplexitet: O(M!·M)

Övriga testfallsgrupper

För övriga testfall måste vi komma på en viktig insikt: det är alltid optimalt för Simon att lägga ut kort med små värden när Nicole lägger ut kort med små värden, och att lägga ut kort med stora värden när Nicole lägger ut kort med stora värden.

Om M=N så vet vi att Simon måste använda **alla** sina kort. Detta, i samband med insikten ovan ger oss att svaret kan fås genom att sortera båda listorna och använda den i:te kortet som Simon med den i:te kortet som Nicole lägger ut.

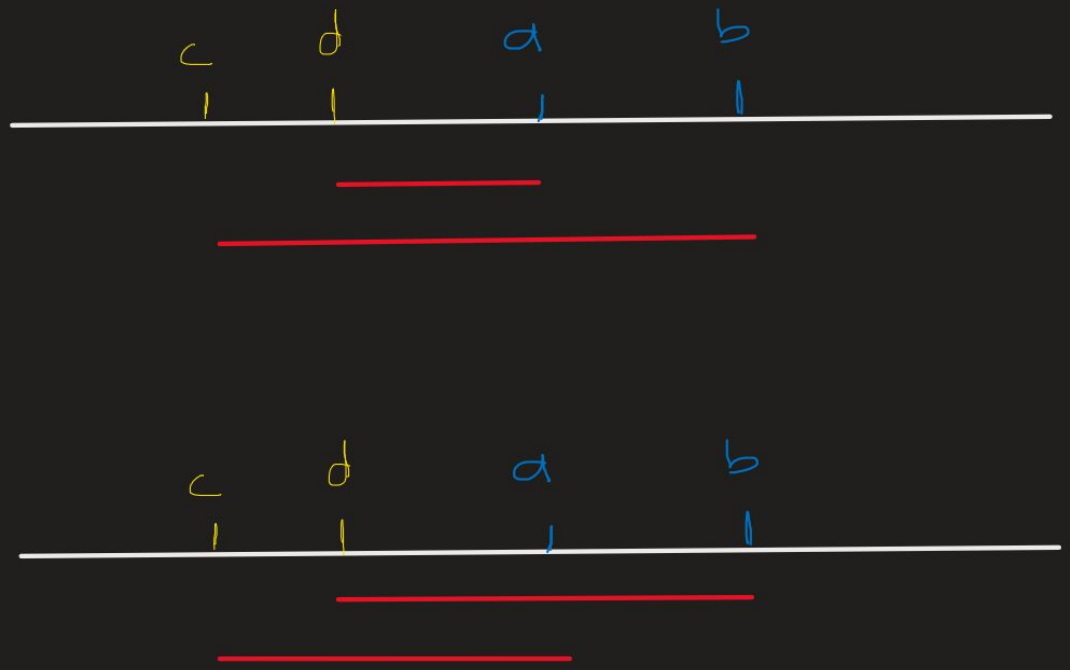
Hur kan man tänka för att komma fram till en sådan insikt och bevisa den? En bra metod är fall uppdelning

Övriga testfallsgrupper

Säg att vi har två kort med tal a och b där a<=b som Nicole lägger ut och Simon vill använda två kort med värden c och d där c<=d. Då finns det tre fall man behöver undersöka (resten av fallen följer av symmetrin): när d<=a, när d >a men c <=d och när c>a men d<=b.

Övriga testfallsgrupper

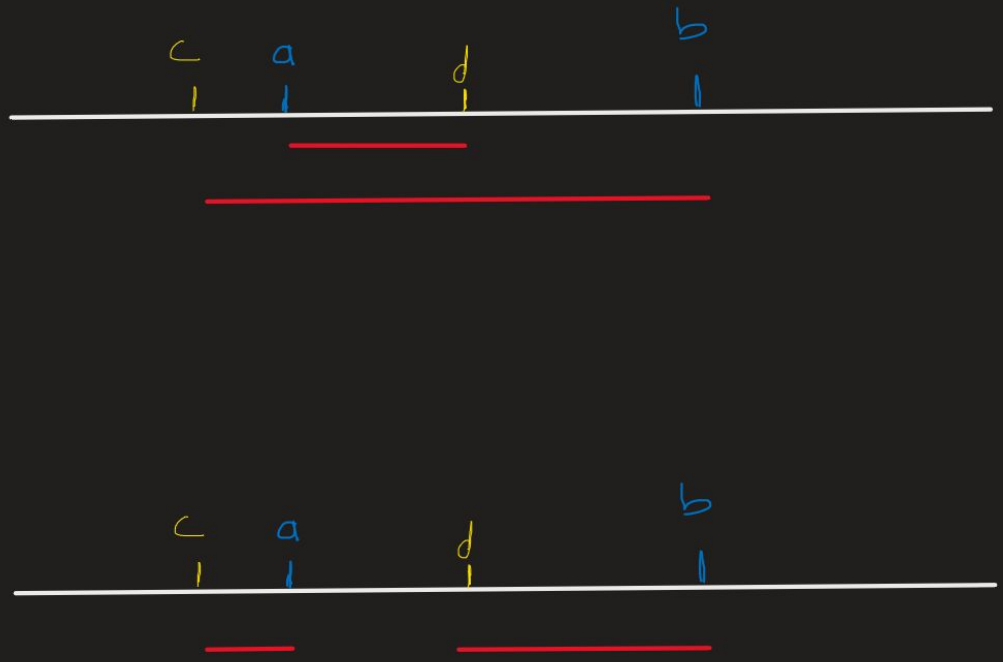
Fall 1 : d<=a



Samma summa, dvs (b-c)+(a-d)

Övriga testfallsgrupper

Fall 2: d>a , c<=a



Summan är (mycket) mindre ifall man kör andra strategin

Övriga testfallsgrupper

Fall 3: a<c<=b<=d

Summan är (mycket) mindre ifall man kör andra strategin

Övriga testfallsgrupper

Nu har vi nämligen löst testfall 3!

Men om M =N+1? Iterera över vilket kort Simon **inte** ska använda.

Man kan räkna det nya resultatet på linjär tid vid varje iteration, då blir tidskomplexiteten O(N^2) vilket är snabb nog för att lösa testfall 2.

För att lösa testfall 4 kan man använda enkel DP eller precomputa på något annat sätt

Tidskomplexiteten: O(N logN+N·(M-N)), och då M-N<=1 så blir tidskomplexiteten O(N logN)

Laserschack

Av: Fredrik Ekholm och Abdullah Zaghmout

Testfallsgrupp 1 (R=1)

Rutnätet består av en rad.

Om det finns rökbomber mellan kungen och attack pjäsen är svaret 1.

Annars är det närmsta rökbomben till kungen eller till attack pjäsen som kommer att förstöra.

Övriga testfallsgrupper

Observationer:

Om avståndet till närmaste rökbomben från en ruta är X så kommer rutan bli fylld med rök efter X sekunder.

Från förra observationen får vi att efter maximalt M+N sekunder kommer Abdullah definitivt inte kunna vinna.

Vi vill kunna snabbt veta när en ruta blir fylld med rök

BFS!

När vi har bestämt vilka rutor som är fyllda med rök vid en viss tidpunkt kan vi köra en vanlig BFS från attack pjäsen och kolla om vi kommer fram till kungen

Övriga testfallsgrupper

Observationer:

Läget blir bara sämre med tiden, dvs så fort Abdullah inte längre kan vinna kan han inte vinna vid en senare tidpunkt.

Huruvida Abdullah kan vinna är monotont. Binärsök!

Grupp	Poängvärde	Gränser
1	10	$R = 1$
2	20	Det finns exakt ett `R` på brädet.
3	20	Ingen ruta är tom (.)
4	20	$R \times C \leq 400$
5	30	Inga ytterligare begränsningar.

Övriga testfallsgrupper

Tid komplexitet O(N·M·log(M+N))

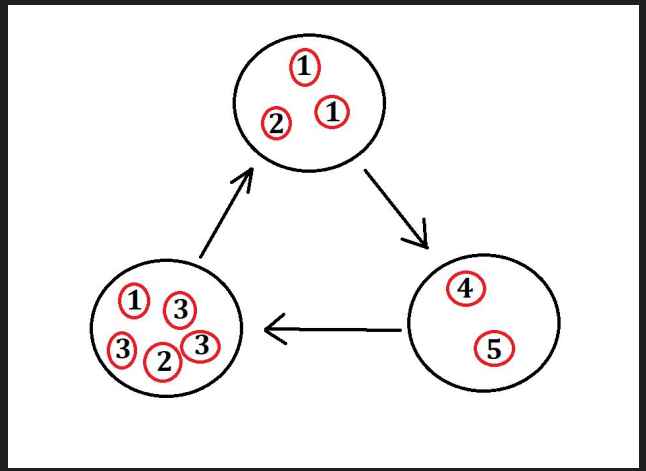
Bonus problem: kan du på en lösning med tidskomplexiteten O(N·M)?

Triangeltal

Av: Nils Gustafsson

Testfall 1-3

1) Om alla A_i är samma så vill vi att den minsta gruppen ska vara så stor som möjligt ⇔ dela in i 3 lika stora grupper och se om det funkar.
2) Om N <= 10 så kan vi testa alla 3^N möjligheter.
3) Om A_i <= 3 så kan vi dela in i tre lika stora grupper om N >= 9, annars kan vi testa alla 3^N möjligheter.

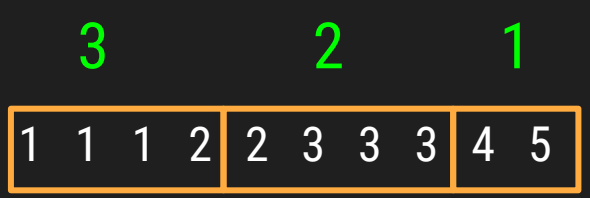


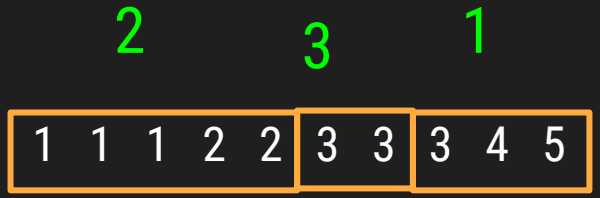
Allmän lösning

Insikt 1: Sortera listan. Det är nu optimalt att låta grupperna vara 3 segment i listan. (Bevis: om $a_i < a_j < a_k$ och i och k tillhör samma grupp men j är i en annan grupp, så kan vi låta i och j byta grupp) . Vi får nu bara $O(N^2)$ möjliga gruppindelningar och kan testa alla för en kvadratisk lösning.

Allmän lösning

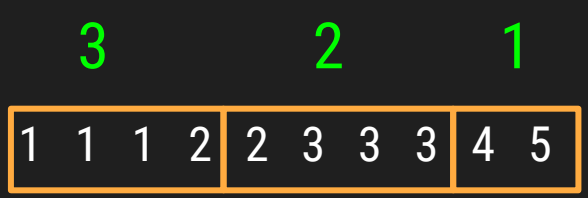
Insikt 1: Sortera listan. Det är nu optimalt att låta grupperna vara 3 segment i listan. (Bevis: om $a_i < a_j < a_k$ och i och k tillhör samma grupp men j är i en annan grupp, så kan vi låta i och j byta grupp) . Vi får nu bara $O(N^2)$ möjliga gruppindelningar och kan testa alla för en kvadratisk lösning.

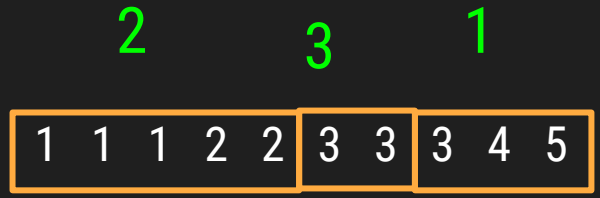




Allmän lösning

Insikt 1: Sortera listan. Det är nu optimalt att låta grupperna vara 3 segment i listan. (Bevis: om $a_i < a_j < a_k$ och i och k tillhör samma grupp men j är i en annan grupp, så kan vi låta i och j byta grupp) . Vi får nu bara $O(N^2)$ möjliga gruppindelningar och kan testa alla för en kvadratisk lösning.





Insikt 2: Det är optimalt att låta grupp i+1 ha storlek $\max(G_i)$, utom möjligen när G_{i+1} innehåller de största elementen. (Bevis: om $|G_{i+1}| > G_i$ så kan vi ta element från G_{i+1} och slänga in i den sista gruppen). Vi får nu bara O(N) möjliga gruppindelningar, t.ex. genom att testa vart största elementet i grupp 2 ska vara.

Triangeltal

Se upp med strängar i Python, **s += “1”** kör i linjär tid!

Utmaning: kan du lösa samma problem med 4 grupper istället?

Öppen fråga: kan du lösa problemet fast med K grupper (helst i O(N * f(K)))?

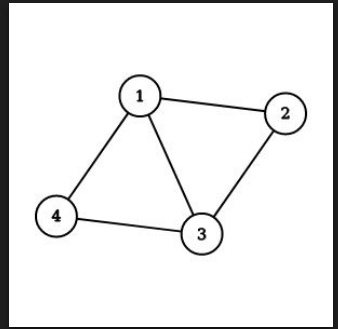
Monopol

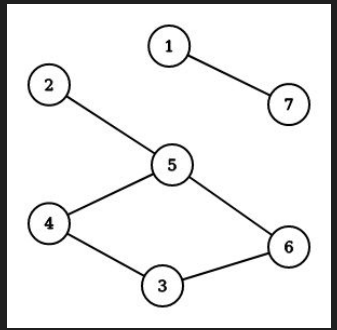
Av: Joakim Blikstad

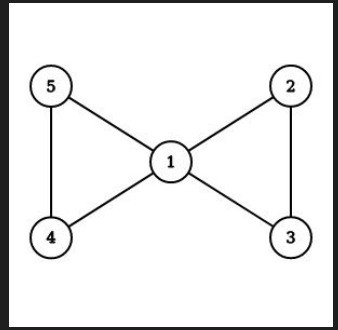
Monopol

Problem: Givet en oriktad graf, hitta en cykel med jämn längd.

Grupp	Poängvärde	Gränser
1	18	$N \leq 10$
2	16	$N \leq 100$ och $M \leq 200$
3	17	Grafen är bipartit
4	13	Alla noder i grafen har grad högst 2
5	20	Alla noder i grafen har grad minst 3
6	16	Inga ytterligare begränsningar



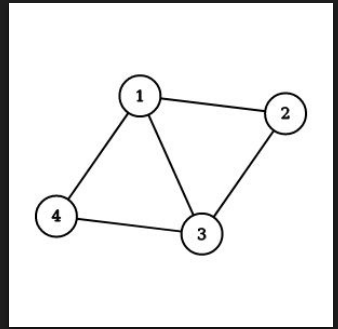


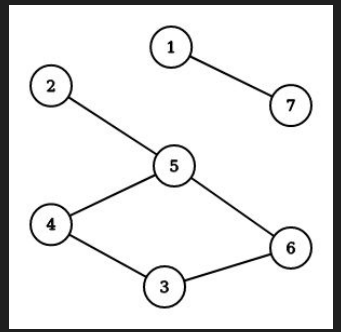


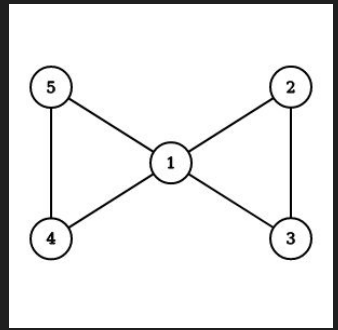
Monopol

Problem: Givet en oriktad graf, hitta en cykel med jämn längd.

	Grupp	Poängvärde	Gränser
Lätt	1	18	$N \leq 10$
	2	16	$N \leq 100$ och $M \leq 200$
Lätt	3	17	Grafen är bipartit
Lätt	4	13	Alla noder i grafen har grad högst 2
	5	20	Alla noder i grafen har grad minst 3
	6	16	Inga ytterligare begränsningar





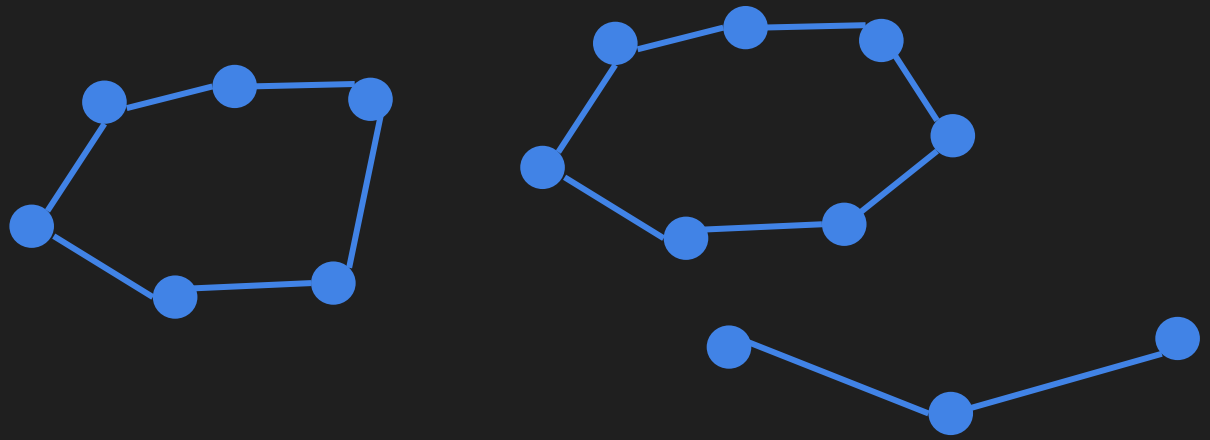


Grupp 1 (N <= 10)

- Prova alla permutationer
- Eller implementera backtracking

Grupp 4 (Noder har grad <= 2)

- Grafen är “linjer” och “cykler” i olika komponenter.
- Kolla varje komponent om den är en cykel med jämn längd

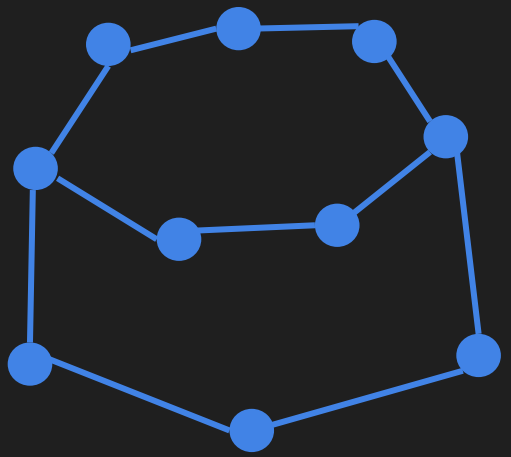


Grupp 3 (Grafen är bipartit)

- Bipartit graf betyder att det inte finns några udda cykler
- Så om vi hittar en cykel så har den jämn längd!
- Kör en DFS i varje komponent:
  - Om vi hittar en kant till en nod som redan är besökt har vi hittat en (jämn) cykel

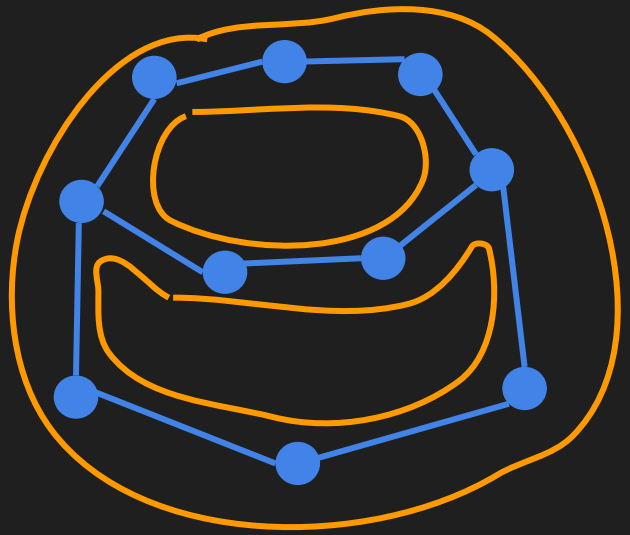
Svårare testfallsgrupper

- Om det finns två cykler som “sitter ihop” (dvs delar minst en kant)



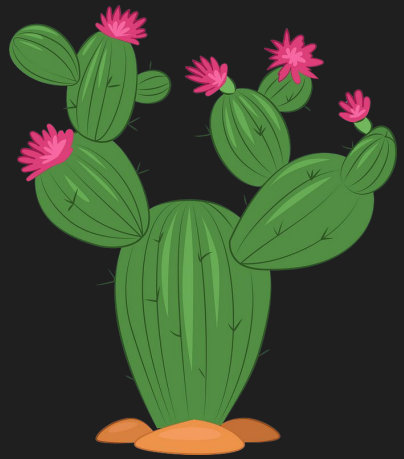
Svårare testfallsgrupper

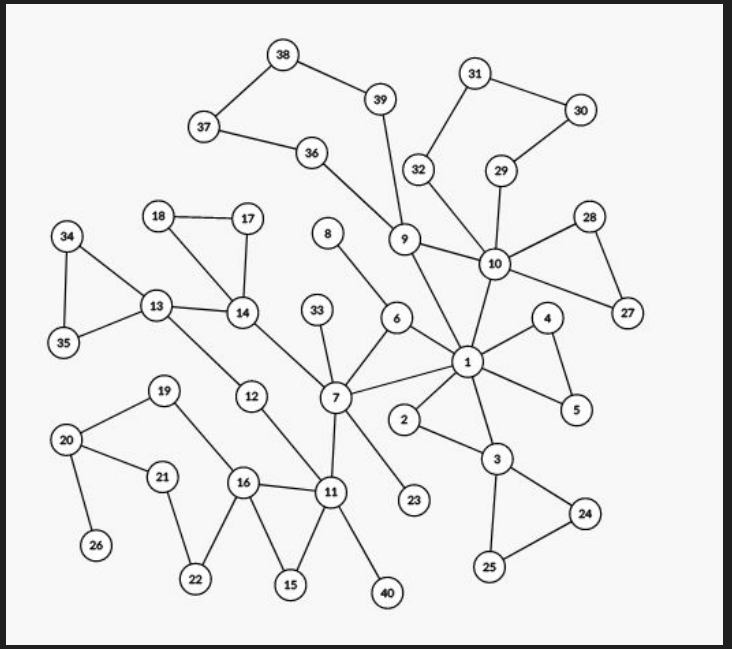
- Om det finns två cykler som “sitter ihop” (dvs delar minst en kant)
- Vi får tre cykler
- Alla kan inte vara udda!
  - Udda + udda = jämnt



Om det inte finns en jämn cykel

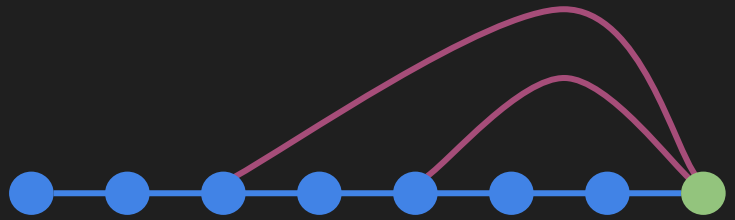
- Träd av udda cykler (“kaktusgraf”)





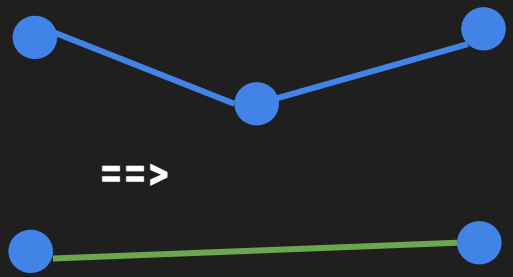
Grupp 5 (Noder har grad >= 3)

- Svaret är alltid ja!
- Hitta en maximal väg
  - DFS tills du hittar en nod v vars alla grannar ligger på din väg
  - v har två till grannar på vägen (grad >= 3)
  - “två cykler som sitter ihop”



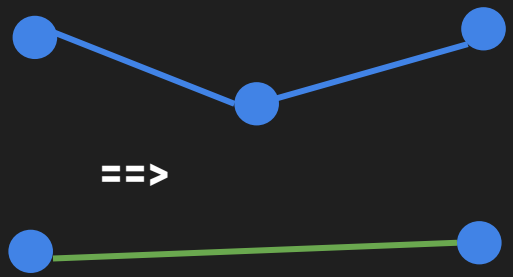
Grupp 1 & 6 (N <= 100 & inga begränsningar)

- Om en nod har grad:
  - <= 1 Ta bort den
  - 2 Byt ut den mot en kant
  - >= 3 Behåll den
- Nu har alla noder grad >= 3



Grupp 1 & 6 (N <= 100 & inga begränsningar)

- Om en nod har grad:
  - <= 1 Ta bort den
  - 2 Byt ut den mot en kant
  - >= 3 Behåll den
- Nu har alla noder grad >= 3



- Implementation lite jobbig -> tidskomplexitet O(N+M)
- Finns många alternativa lösningar
  - Slumpad DFS har en sannolikhet >= 1/2 att hitta en jämn cykeln om någon finns
    - Övning: bevisa detta :)

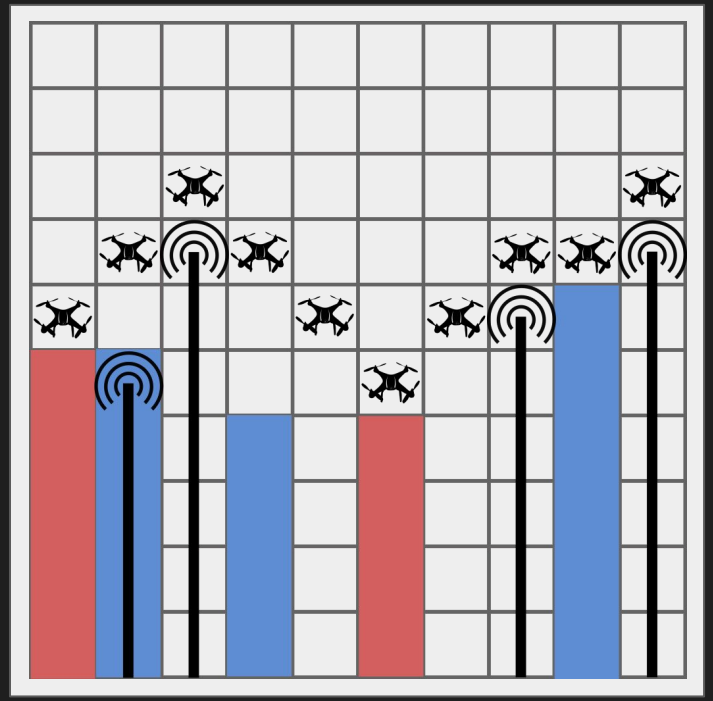
Legobyggartävlingen

Av: Mattias Akke, Simon Lindholm och Fredrik Ekholm

Legobyggartävlingen

- Ta bort några master för att maximera din poäng
- Drönarna “hänger” från masterna och förstör kapar av torn de krockar med
- Poäng på ett torn är “höjd × f_i”

Grupp	Poängvärde	Gränser
1	23	Masterna har lägre x-koordinat än tornen
2	10	$A, B, M \leq 8$
3	22	$A, B, M \leq 100$
4	45	Inga ytterligare begränsningar



Grupp 1 (Alla master till vänster om alla torn)

- Finns alltid en optimal lösning som behåller endast en mast
  - Prova alla!
- Räkna ut poängen med att för varje torn kolla hur högt det blir
  - Drönar-höjd på koordinat x är
    - h_i + x_i - x om vi valde mast (x_i, h_i)

Grupp 3 (antal torn och master <= 8)

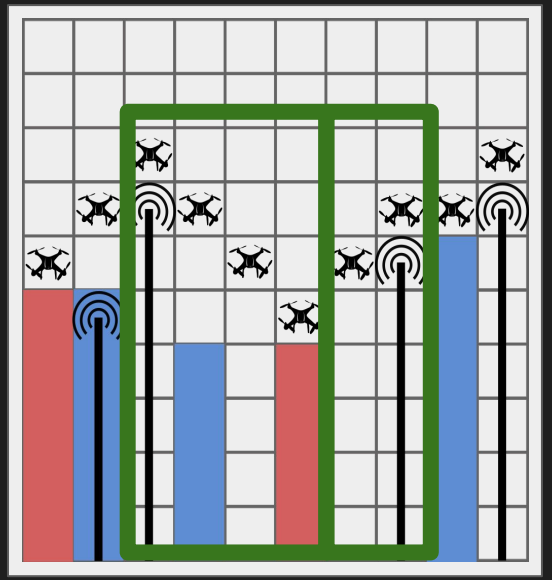
- Prova att behåll alla delmänger av masterna
- Räkna ut poängen med att för varje torn kolla hur högt det blir
  - Drönar-höjd på koordinat x är
    - max { h_i - |x_i - x| } över all master (x_i, h_i)
- Tidskomplexitet O(2^{master} * master * torn)

Grupp 2 (antal torn och master <= 100)

- Dynamisk programmering
- Sortera masterna efter x-koordinat
- DP[k] = maximala poängen till vänster om mast k
  givet att drönaren passerar precis ovanför mast k
- DP[k] = max_{j < k} (DP[j] + cost(j,k))
  - cost(j,k) är poängen man får mellan mast j och k om man väljer dessa master
  - Går att beräkna cost(j,k) O(torn)
- Tidskomplexitet O(master^2 * torn)

Full lösning

- Samma DP, men vi vill beräkna cost(j,k) snabbare
  - cost(j,k) är poängen man får mellan mast j och k om man väljer dessa master
- Alternativ 1:
  - Datastrukturer (eg 2D-segmentträd) för att querya cost(j,k)
  - Tidskomplexitet $O(master^2 * \log^2(max\_x\_coordinate)$
- Alternativ 2:
  - För varje mast, förberäkna till höger och vänster kostnaden om man drönaren flyger diagonalt nedåt s steg
  - cost(j,k) = cost j to lowest point + cost lowest point to k
  - Tidskomplexitet $O(master^2 + master * torn)$
    eller $O(master^2 + master * max\_höjd + torn)$
- Halvknepig implementation...



IOI-uttagningen

Internationella mästerskap

- IOI
  - Top 4 i uttagningen
  - Oklart datum, oklart online/onsite/baltisk samling
- BOI
  - Top 4 i uttagning + 2 till (oftast, men inte alltid, 2 bästa icke-treorna)
  - Oklart datum, kanske senarelagt och onsite

Uttagningsmoment

- Finalen (100)
- Lägret (100)
  - 13 februari (5h på morgonen)
- KATT (100)
  - 21-27 feb (under valfria 24h)
- Nordiska (100)
  - 16 mars (5h under skoltid)
- Teori 1 (50)
  - Fram till söndag 6 feb
- Teori 2 (50)
  - 28 feb till 13 mars
- Alla moment summeras för uttagningen - poängen skalas om (så 600p i finalen = 100p)

Teoriblad

- 4 svåra problem som löses på papper. Inget kodande krävs, endast pseudokod.
- Detaljer kring bedömning etc står på teoribladet
- Inlämning via mail

Upsolving

- Samtliga kodtävlingar kan **fram tills nästa tävling** *upsolvas*
- Ni kan fortsätta skicka in lösningar på kattis
- Om ni efter 1 vecka har X poäng och vid tävlingsslut har Y poäng får ni (X - Y) / 4 extrapoäng
- Dvs: nollade ni ett problem kan ni ändå höja er totalpoäng med 25 för uttagningen!
- Varje tävling har lösningsgenomgång, och ni får diskutera hur man löser problemen med varandra (och be om hjälp på discord), men ni måste *koda själva* - d.v.s. inte hjälpa varandra med själva implementeringen

Andra tävlingar

- Codingcup: En serie tävlingar som vanligtvis ordnas på olika ställen i Sverige. PO-finalen är en av deltävlingarna. https://codingcup.se/
- ICPC: Den största universitetstävlingen, där man deltar i lag med 1-3 personer. Drar igång med NCPC på hösten.
- Code Jam: Googles årliga tävling, brukar vara på våren.
- Hackercup: Facebooks motsvarighet.

Onlinedomare

- codeforces.com: Ordnar tävlingar ganska regelbundet (~en gång i veckan). Har även ett forum.
- atcoder.jp: Har också tävlingar ungefär en gång i veckan. Har ofta mer matematiska problem.

Kodsport

- http://www.kodsport.se/ -> Bli medlem!
- Helt gratis att gå med
- Nyhetsbrev om våra aktiviteter
- Mer bidragspengar från vårt moderförbund, Unga Forskare