Vinjetter
TDDC91 Datastrukturer och algoritmer

28 augusti 2017

Scenario 1 (Obligatoriskt)

Man har inom Posten Logistik AB skrivit programvara för sortering av kundinformation och vill standardisera användningen av sorteringsalgoritmer. Som ett första steg har man provkört de olika algoritmerna på samma dator. Man har provat data av olika storlek och uppnått följande resultat:

storlek indata	alg. 1	alg. 2	alg. 3	alg. 4
10	0.00044	0.00057	0.00041	0.00046
100	0.00675	0.00420	0.00171	0.00244
1000	0.59564	0.05565	0.02927	0.02587
10000	58.864	0.71650	0.42998	0.31532
100000	∞	8.8591	5.7298	3.5882
1000000	∞	104.68	71.164	41.282

Scenario 2 (Obligatoriskt)

ADT Map/Dictionary har många tillämpningar. Följande är några exempel:

1. Uppslagstabell för att beräkna $\sin(\theta)$, där θ är en av 1000000 möjliga vinklar jämnt fördelade mellan 0 och π.
2. Databas som avbildar ljuddata (från en fil) på artistnamn.
3. Snabbaste garantierna för insättning, borttagning och sökning för en godtycklig uppsättning numeriskt data.

Spelar det någon roll vilken underliggande implementation man väljer i ovanstående tllämpningar?

Scenario 3 (Valbart)

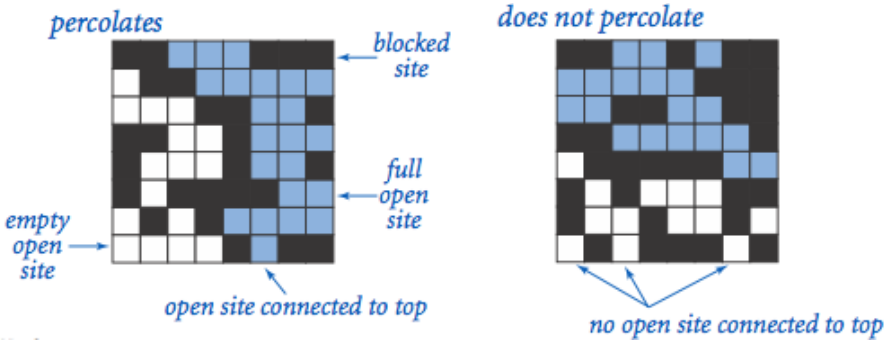
Försök att hitta en uppskattning av värdet på *perkolationströskeln*.

Tillämpningar

Givet ett sammansatt system bestående av slumpvis fördelade isolerande och metalliska material: hur stor andel av materialen behöver vara metalliska för att systemet som helhet skall vara en elektrisk ledare? Givet ett poröst landskap med vatten på ytan (eller olja under), under vilka omständigheter kan vattnet tränga igenom till bottnen (eller oljan tränga upp genom ytan)? Vetenskapen har tagit fram den abstrakta processen *perkolation* för att modellera sådana situationer.

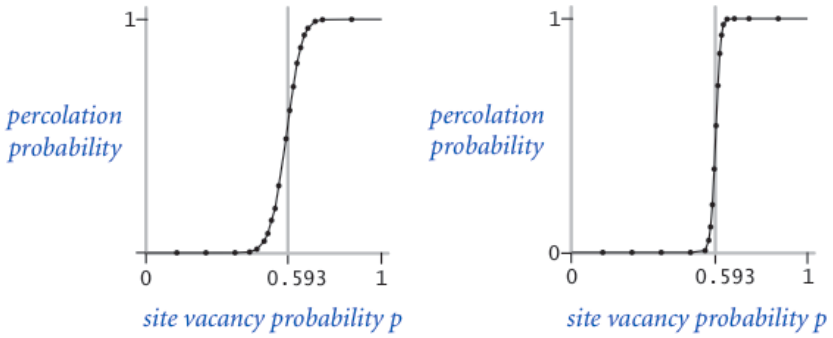
Modellen och problemet

Vi modellerar ett perkolationssystem som ett $N \times N$-rutnät av *platser*. Varje plats är antingen *öppen* eller *blockerad*. En *full* plats är en öppen plats som kan bindas samman till en öppen plats i övre raden via en kedja av närliggande (vänster, höger, upp, ner) öppna platser. Vi säger att systemet *perkolerar* om det finns en full plats på bottenraden. Med andra ord, ett system perkolerar om vi fyller alla öppna platser i översta raden och den processen fyller någon öppen plats i bottenraden. (I materialexemplet är de öppna platserna metalliska material och i det porösa landskapet motsvarar de öppna platserna hålrum genom vilket vatten kan flöda.)



Följande har länge varit en öppen fråga: Om vi låter platser vara öppna oberoende av varandra med sannolikhet p (och därför blockerade med sannolikhet $1 - p$), vad är sannolikheten att systemet perkolerar? När p är 0 perkolerar inte

systemet; när p är 1 perkolerar systemet. Bilden nedan visar perkolationssannolikheten som funktion av p för slumpmässiga 20×20-rutnät (vänster) och slumpmässiga 100×100-rutnät (höger).



När N är tillräckligt stort finns ett *tröskelvärde* p^* sådant att när $p < p^*$ perkolerar ett slumpmissigt $N \times N$-rutnät nästan aldrig och när $p > p^*$ perkolerar ett slumpmässigt $N \times N$-rutnät nästan alltid. Ingen har hittills lyckats härleda en analytisk lösning som bestämmer värdet på perkolationströskeln p^*.

Monte Carlo-simulering

Betrakta följande beräkningsexperiment för att uppskatta perkolationströskeln:

- Låt alla platser vara blockerade.
- Upprepa följande till systemet perkolerar:
  - Välj en plats (rad i, kolumn j) med likformig sannolikhet bland alla blockerade platser.
  - Öppna plats (rad i, kolumn j).
- Andelen platser som är öppna när systemet perkolerar är en uppskattning av perkolationströskeln.

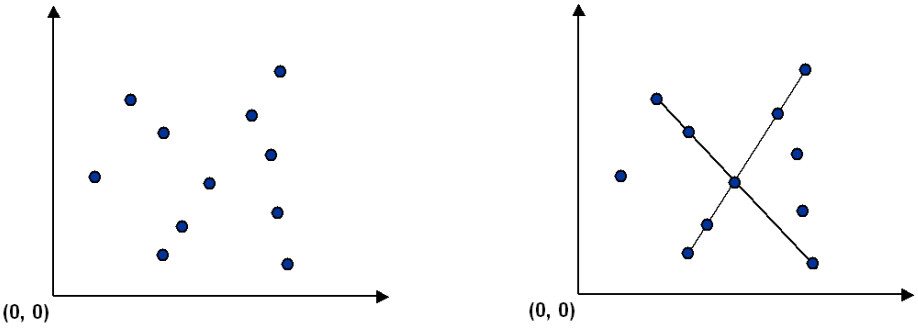
Genom att upprepa ovanstående experiment T gånger och ta medelvärdet får vi en bättre uppskattning av perkolationströskeln.

Exempelprogram och data

På katalogen /home/TDDC91/vinjetter/percolation/ finns ett program som modellerar ett perkolationssystem (`Percolation.java`), men det fattas en viktig bit. Vidare finns programmet `PercolationStats.java` som använder modellen för att utföra Monte Carlo-simuleringen som skissats ovan. Slutligen finns också `PercolationVisualizer.java`, med tillhörande datafiler och stödprogram, för att visualisera ett perkoleringsförlopp.

Scenario 4 (Obligatoriskt)

Givet en mängd av `N` distinkta punkter i planet, hitta alla (maximala) linjesegment som innehåller en delmängd av 4 eller fler av punkterna.



Tillämpningar

Viktiga komponenter i datorseende är att använda mönsteranalys av bilder för att rekonstruera de verkliga objekt som genererat bilderna. Denna process delas ofta upp i två faser: *feature detection* och *pattern recognition*. I *feature detection* väljs viktiga områden hos bilden ut; i *pattern recognition* försöker man känna igen mönster i områdena. Här får ni chansen att undersöka ett särskilt rent mönsterigenkänningsproblem rörande punkter och linjesegment. Den här typen av mönsterigenkänning dyker upp i många andra tillämpningar som t.ex. statistisk dataanalys.

Exempeldata och -program

På katalogen /home/TDDC91/vinjetter/pattern/ finns flera filer med exempeldata. Där finns också programmet `Brute.java` som löser mönsterigenkänningsproblemet.

Scenario 5 (Obligatoriskt)

Dagbrott AB vill bryta malm för att maximera sin nettovinst. Området där dagbrottet ska anläggas är indelat i block och för enkelhets skull antar vi en endimensionell layout. Från den geologiska prospekteringen är bedömningen att brytning av block `i` ger nettovinst (värdet av malmen minus utvinningskostnaderna) a_i miljoner kronor. Vissa miljörelaterade restriktioner gör att Dagbrott AB endast får bryta malm i en sammanhängande grupp av block. Vilka block bör Dagbrott AB välja? Med andra ord, givet en sekvens av `N` (möjligen negativa) heltal, bestäm den största möjliga summan bland alla sammanhängande sekvenser. Problemet är lätt om alla tal är positiva: välj hela sekvensen. Svårigheten är när det förekommer negativa tal: borde man ta med ett negativt tal i hopp om att närliggande positiva tal kompenserar för detta?

För indatat nedan är den maximalt möjliga summan 104, vilken erhålls genom att ta med block 2 till 6. Notera att summan alltid är minst 0, eftersom Dagbrott AB kan välja att inte gräva alls.

block	0	1	2	3	4	5	6	7	8	9	10
nettovinst	12	-34	40	6	-10	56	12	-1	-15	10	4

Tillämpningar

I verkliga tillämpningar behöver gruvoperatörer lösa en tredimensionell variant av problemet. Dessutom kan det förekomma ytterligare geologiska begränsningar, t.ex. att det inte går att bryta ett område 100 meter under marken utan att först frilägga några av de omgivande områdena. Den tvådimensionella varianten av problemet förekommer även som del av bildbehandling, där målet är att bestämma maximum likelihood-skattningen för ett visst sorts mönster.

Exempeldata

På katalogen /home/TDDC91/vinjetter/pitmining/ finns flera filer med exempeldata.