DD2458, Problemlösning och programmering under press

Föreläsning 10: Kombinatorik

Datum: 2009-11-18
Skribenter: Cecilia Roes, Ann-Sofie Lindblom, Ollanta Cuba Gyllensten
Föreläsare: Fredrik Niemelä

1 Delmängder

En delmängd av en mängd $[n] = a_1, a_2, ..., a_n$ är en mängd sådan att varje element i delmängden finns i mängden $[n]$. För att konstruera en delmängd kan man för varje element i ursprungsmängden bestämma huruvida det ska vara en del av delmängden eller inte, det totala antalet delmängder blir då 2^n, alltså antalet möjliga kombinationer av n sådana val.

1.1 Binomialtalen

$\binom{n}{k}$ kallas binomialtal och är antalet delmängder av n med storlek k. Man kan också se det som antalet sätt att välja k element ur n. Notera att $\binom{n}{k}$ bara är definierat för icke-negativa k och n, och för $k > n$ är $\binom{n}{k}$ noll, eftersom det inte finns något sätt att välja fler element än vad som finns i mängden. Totala antalet delmängder är då summan av alla binomialtal där k går från 0 till n:

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n$$

Binomialtalen kan definieras rekursivt på följande sätt

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \binom{n}{0} = \binom{n}{n} = 1.$$

Detta kan visas genom konstruktion på följande sätt:

- Välj ett godtyckligt element x i en mängd X av storlek n, för varje delmängd av storlek k gäller att detta element antingen är i delmängden, eller i delmängdens komplement.
- Antalet delmängder (av storlek k) sådana att elementet är i delmängden är antalet delmängder av storlek $k-1$ ur mängden $X \setminus \{x\}$, detta blir $\binom{n-1}{k-1}$.
- Antalet delmängder sådana att x är i dess komplement är helt enkelt antalet delmängder av storlek k ur $X \setminus \{x\}$, alltså $\binom{n-1}{k}$.

Binomialtalen är kopplade till Pascals triangel, då denna byggs upp på ett liknande sätt. I triangeln är värdet på nuvarande position summan av de två närmaste talen på raden ovanför. Så för att hitta $\binom{n}{k}$ i Pascals triangel tittar man på det k:te elementet på rad n.

```
                1
            1       1
        1       2       1
    1       3       3       1
1       4       6       4       1
```

Figure 1: Pascals triangel

Man kan beräkna binomialtalen direkt ur den rekursiva formeln ovan. Görs detta utan memoisering är tidskomplexiteten exponentiell, $O(2^n)$. Med memoisering däremot får vi $O(n^2)$, då talen i figur 2 måste beräknas enbart en gång.

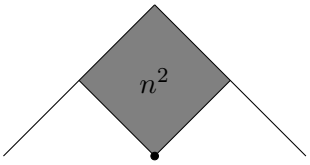

Figure 2: Beräknade binomialtal

För att beräkningen skall bli mer effektiv kan man istället använda formeln:

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

$$\frac{1 \cdot 2 \cdot \ldots \cdot (n-k) \cdot (n-k+1) \cdot \ldots \cdot n}{(1 \cdot 2 \cdot \ldots \cdot (n-k))(1 \cdot 2 \cdot \ldots \cdot k)} = \frac{(n-k+1) \cdot (n-k+2) \cdot \ldots \cdot n}{1 \cdot 2 \cdot \ldots \cdot k},$$

vilket ger oss algoritm 1.

Algorithm 1: Räknar ut $\binom{n}{k}$

```
BINOMIAL(n, k)
(1)   r ← 1
(2)   for i = 1 to k
(3)       r = r · (n − k + i)/i
(4)   return r
```

Att vi i for-loopen kan dela med i beror på att hela tiden är minst en av termerna vi multiplicerat in i r delbar med i. För att detta skall vara sant måste vi först multiplicera innan vi delar.

Denna algoritm går i $O(n)$, men har lite större risk för overflow. Om man använder den rekursiva formeln är största talet utdata, men i den här algoritmen får vi i sista varvet av for-loopen ett tal som är en faktor k större än svaret.

För att generera alla delmängder av en mängd med storlek n kan man helt enkelt räkna uppåt från 0 till 2^n och titta på den binära representationen av dessa tal, där en etta på position i betyder att det i:te elementet är med i delmängden, och en nolla betyder att det inte är det.

1.2 Multinomialtalen

$\binom{n}{n_1,n_2,\ldots,n_k}$ kallas multinomialtal och är antalet sätt man kan dela en mängd av storlek n i delmängder av storlekar $n_1, n_2, \ldots, n_k$ där $\sum_{i=1}^{k} n_i = n$. Likt binomialtalen kan multinomialtalen definieras rekursivt som

$$\binom{n}{n_1, n_2, \ldots, n_k} = \sum_{i=1}^{k} \binom{n-1}{n_1, n_2, \ldots, n_i - 1, \ldots, n_k} = \frac{n!}{n_1! n_2! \cdots n_k!}.$$

Alternativt kan de definieras ur binomialtalen som

$$\binom{n}{n_1, n_2, \ldots, n_k} = \prod_{i=1}^{k} \binom{n - \sum_{j=1}^{i-1} n_j}{n_i}$$

, då en motsvarande uppdelning kan konstrueras genom att först välja n_1 element ur den ursprungliga mängden, sedan n_2 ur den resterande och så vidare.

Notera att det sista elementet i varje sådan expansion alltid är lika med 1. Detta ger ett enkelt sätt att återanvända algoritmer för binomialtal för att beräkna multinomialtal. Man ser även lätt ur denna identitet att multinomialtalet $\binom{n}{k,n-k}$ är lika med binomialtalet $\binom{n}{k}$.

2 Permutationer

Om man bara har unika element är antalet permutationer av n element $n!$ stycken, först väljs ett element bland n, sen ett bland $n-1$ osv. Hur många finns det då man tillåter upprepningar?

> *Exempel:* Hur många permutationer finns det av strängen "abacca"?
> Elementen a^3bc^2 ska placeras ut på 6 platser. Att placera ut 3 stycken a:n på 6 platser kan göras på $\binom{6}{3}$ sätt. Det lämnar 3 platser kvar till b, som alltså kan sättas ut på $\binom{3}{1}$ sätt. Sedan finns det 2 platser kvar till de 2 c:na $\binom{2}{2}$. Detta är samma sak som antalet delmängder där vi har 3,2 resp 1 element i varje $\binom{6}{3,2,1}$. Om man tänker sig att man delar upp positionerna, istället för bokstäverna, i delmängder så ser man att man får 3 delmängder, en med 3 element där a:na skall få plats, en med 2 element för c:na och en med 1 element för b:et.

Antalet permutationer när man tillåter upprepningar är alltså multinomialtalet $\binom{n}{n_1,n_2,\ldots,n_k}$ där n_i är antalet förekomster av element i.

Hur gör man då för att generera alla permutationer? Det gör man med 3 enkla steg:

1. Hitta första
2. Ta nästa, upprepa
3. Hitta sista

Första elementet är den sorterade strängen, tex "aaabcc", och sista är den bakåtsorterade strängen "ccbaaa".

Hur hittar man då nästa? Det är alltid en fråga om att byta plats på saker, och man vill alltid byta så lite som möjligt, den nästföljande strängen skall vara så lite större än den föregående som möjligt. Man vill byta elementet på den minst signifikanta platsen som är mindre än något av elementen till höger om sig och man vill byta med det minsta element till höger som är större. Sedan vill man sortera allt till höger för att få en så liten ökning som möjligt. Ex. abac|**ac**| → abac|**ca**| och ac|**acb**a| → ac|**bca**a| → ac|**baa**c|. Om elementen inte har en inbördes ordning kan en godtycklig ordning användas.

Algorithm 2: Hittar nästa permutation

```
NEXT_PERMUTATION(x, n)
(1)  for i = n − 1 to 1
(2)      if x_i < x_{i+1}
(3)          break
(4)  for j = n to i + 1
(5)      if x_j > x_i
(6)          break
(7)  swap(x_j,x_i)
(8)  reverse(x_{i+1},x_n)
(9)  return x
```

Permutationsalgoritmen kan även användas till att generera alla delmängder med k element. Man skapar då en sträng bestående av k stycken 1:or och $n-k$ 0:or och kör permutationsalgoritmen på den.

För binära tal finns det en bättre algoritm, som kan användas på system som använder 2-komplements representation.

Algorithm 3: Hittar nästa permutation

```
NEXT(n)
(1)  c ← n AND −n                          % "find j"
(2)  r ← n + c                             % swap
(3)  return ((r XOR n)/(4 · c)) OR r       % reverse
```

Ett alternativ för portabel kod är att använda prefixinkrement istället för bitoperationerna.

3 Partitioner

En partition av en mängd, är en uppdelning av mängden i disjunkta, icketomma delmängder, dvs.

$B = B_1, B_2, \ldots, B_k$ är en partitionering i k delar omm

- $\bigcup B_i = [n]$
- $B_i \bigcap B_j = \emptyset$
- $B_i \neq \emptyset$

Dessa delmängder kallas även block.

3.1 Stirlingtal

Antalet partitioner av $[n]$ i k block är Stirlingtalen av det andra slaget, dessa kan definieras rekursivt som

$$S(n,k) = S(n-1,k-1) + k \cdot S(n-1,k),$$

$$S(n,k) = \begin{cases} 0 & k > n, k < 0, n < 0 \\ 1 & k = 0, n = 0 \end{cases}$$

Liksom med binomialtal kan detta visas genom konstruktion:

- Välj ett godtyckligt element x ur en mängd X med storlek n, detta element är antingen ensam i sitt block, eller inte.
- Av det första fallet måste det finnas $S(n-1,k-1)$ kombinationer, dvs. antalet partitioner av $X \setminus \{x\}$ i $k-1$ block.
- För att beräkna antalet kombinationer av det andra fallet konstateras att $X \setminus \{x\}$ kan partitioneras i k block på $S(n-1,k)$ olika sätt, i varje sådan kombination kan k olika partitioner av X bildas genom att "stoppa in" x i de k olika blocken. Det totala antalet partitioner i det andra fallet blir alltså $k \cdot S(n-1,k)$.

3.2 Belltal

Belltalen betecknar det totala antalet partitioner av en mängd med storlek n. Dessa har ett enkelt samband till Stirlingtalen, då det är summan av antalet partitioner av mängden i i block för alla $1 \leq i \leq n$.

$$B(n) = \sum_{k=1}^{n} S(n,k)$$

Belltalen kan även definieras rekursivt som

$$B(n) = \sum_{k=1}^{n} \binom{n-1}{k-1} B(n-k).$$

Vilket kan visas genom konstruktion som följer:

- Välj ett godtyckligt element x ur en mängd X av storlek n. I varje partition av mängden X är x i en partition, denna partition har en storlek k, $1 \leq k \leq n$.
- Antalet sätt att konstruera en partition av storlek k ur mängden $X \setminus \{x\}$ är $\binom{n-1}{k-1}$.
- Antalet sätt att partitionera resterande $n-k$ element på är $B(n-k)$. Antalet partitioner av X sådan att x tillhör en partition av storlek k är alltså

$$\binom{n-1}{k-1} \cdot B(n-k)$$

- Det totala antalet sätt blir då $\displaystyle\sum_{k=1}^{n} \binom{n-1}{k-1} B(n-k)$.

4 Catalantal

Catalantalen kan användas till mycket, de beskriver bland annat antalet binära träd och antalet sätt att korrekt placera ut parenteser. Talen kan definieras rekursivt som

$$C(n) = \sum_{k=0}^{n-1} C(k) \cdot C(n-k-1), C(0) = 1,$$

där man kan tänka sig att man summerar över alla kombinationer av storlekar på delträden, där k är storleken på det vänstra delträdet, och $n-k-1$ blir storleken på högra delträdet. Alla kombinationer för $C(3)$ visas i figuren nedan.

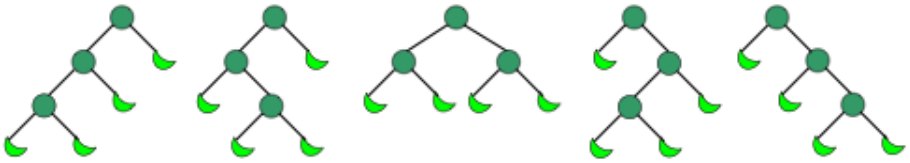

Figure 3: Catalantal som träd

Parallellen med paranteser kan förklaras genom att en korrekt parentetisering alltid kan skrivas på formen $(X)Y$, där X motsvarar vänster delträd och Y motsvarar höger delträd (dessa är alltså även de på formen $(X')Y'$ såtillvida de inte är tomma, i vilket fall de representeras av den tomma strängen.