$\pi\mathrm{l}\alpha y\sqrt{\mathbf{mat}\chi}$

Kombinatorni algoritmi

Goran Žužić

U informatici se često pojavljuju zadatci tipa koliko ima različitih objekata koji zadovoljavaju neke uvjete. Pristup rješavanja takvih zadataka je specifičan pa je njihova riješenost vrlo niska. Ovdje će biti prikazani neki od najčešćih načina rješavanja.

Osnovna ideja

Pretpostavimo da imamo **dovoljno vremena** za generiranje svih mogućih objekata pojedinačno. Traženi objekt obično se može raščlaniti na podobjekte (npr. riječi na slova) koje pojedinačno fiksiramo i tako generiramo ono što se od nas traži.

Primjer 1. Ispiši sve riječi duljine dva koristeći samo mala slova engleske abecede.

Rješenje. Recimo da na početku imamo dvije prazne kućice u koje upisujemo slova. Prvo ćemo u prvu kućicu upisati slovo 'a' i onda generirati sve riječi duljine 1 na koje ćemo s lijeve strane prilijepiti slovo a. Potom ćemo isto napraviti za b i tako dalje...

a	a
a	b
a	c
a	d
⋮	⋮

g	i
g	j
⋮	⋮
z	y
z	z

Ovaj način generiranja lako bismo mogli napisati u pseudokodu...

procedura generiraj(duljina, riječ S);
ako *duljina=0*
  ispiši(S);
  prekini proceduru;
u suprotnom
  za *c='a' do 'z'* **radi**
    generiraj(duljina−1, S+c)

✓

Naš je zadatak na primjer bio da prebrojimo broj takvih riječi (bez formule). Mogli bismo, naravno, umjesto ispisa staviti neki brojač koji bi brojao takve riječi, ali to bi za veće vrijednosti tražene duljine bilo nemoguće zbog veličine rješenja.

Da radimo s brojačem, primijetili bismo da funkcija generiraj stvara isti broj riječi ovisno o argumentu duljini (generiraj(1,S) uvijek generira 26 riječi), pa se ona poziva sa istim argumentom više puta. Ovaj problem višestrukog izračunavanja istih vrijednosti nalazi se u članku o dinamičkom programiranju.[1]. Prije nego što krenete dalje čitati, bilo bi preporučljivo da pročitate taj članak.

Broj riječi u proceduri generiraj ne ovisi o već fiksiranim slovima, pa slova uopće nemaju utjecaja na krajnji rezultat. Zbog toga bi dinamičko rješenje ovog zadatka izgledalo ovako:

$F[0] := 1$;
za $I = 1$ *do duljina* **radi**
  za *c='a' do 'z'* **radi**
    $F[I] = F[I] + F[I-1]$;

Isto tako znamo da postoji 26 ravnopravnih slova u engleskoj abecedi:

[1] *Ivo Sluganović:* Dinamičko programiranje, *PlayMath* br. 7 (2005.)

$$\pi \mathbf{l} \alpha y \sqrt{\mathbf{mat}\chi}$$

$F[0] = 1;$
za $I = 1$ *do duljina* **radi**
  $F[I] = 26 \cdot F[I-1]$

Ovo nas opet dovodi do izravne formule koja glasi 26^{duljina}...

Primjer 2. Koliko ima riječi duljine manje od 20 slova u kojima su sva slova zapisana u neopadajućem poretku ako gledamo englesku abecedu (npr. "g", "azz" ili "abcz")?

Rješenje. Primijetit ćemo da broj generiranih riječi u generatoru više ne ovisi samo o preostaloj duljini, nego i o zadnjem slovu. Te su informacije sve što trebamo pamtiti da bismo izračunali traženi broj. Algoritam izgleda ovako:

```
funkcija prebroji(duljina, zadnje_slovo)
        ako smo prije našli vrijednost za argumente (duljina, zadnje_slovo) nemamo što računati

        suma = 0
        za svako slovo c od zadnje_slovo do 'z'
                        suma = suma + prebroji(duljina-1, c)
        zapamti u matrici vrijednost suma za argumente (duljina, zadnje_slovo)
        vrati suma
```

Traženu vrijednost dobit ćemo ako pozovemo prebroji(duljina, 'a'). Sada možemo primijetiti da je ovakav problem jako teško analitički riješiti matematičkim metodama, puno je lakše rješiv dinamičkim programiranjem.

Ovakav algoritam vrlo je efikasan jer će se funkcija prebroji izvršavati najviše $20 \cdot 26$ puta (višestruki pozivi sa istim argumentom odmah će se prekinuti zbog memoizacije rješenja), a pritom se petlja izvršiti najviše 26 puta, što nam daje najveću složenost od $20 \cdot 26 \cdot 26 = 13520$ operacija (nekoliko nanosekundi na današnjem računalu). ✓

Argumente funkcije koja nam služi za prebrojavanje objekata u daljnjem ćemo tekstu zvati **stanje dinamičkog rješenja**. Dakle, stanje u ovom problemu opisano je duljinom i zadnjim slovom u riječi.

Primjer 3. Na koliko se načina može od N objekata izabrati njih K? ($N \leq 20$, $K \leq N$)

Rješenje. Lako možemo shvatiti da je rješenje ovoga ekvivalentno ***binomnom koeficijentu*** $\binom{N}{K}$ koji je jednak[2]:

$$\binom{N}{K} = \frac{N!}{(N-K)!K!}.$$

Ovakvu formulu možemo lako i efikasno izračunati, ali će se pritom javiti problem preciznosti jer će srednje vrijednosti koje dobivamo računanjem daleko premašiti okvir podataka kojima raspolažemo iako će krajnji rezultat biti relativno malen ($20! > 10^{18}$). Stoga ćemo pokušati naći drugi način rješavanja.

Kada pokušavamo izračunati $\binom{N}{K}$, promotrimo N-ti objekt. Postoje dvije mogućnosti: ili ćemo ga uzeti ili ga nećemo uzeti. Te se mogućnosti ne preklapaju, a ujedno pokrivaju sva moguća rješenja. Ukoliko ga uzmemo, onda još moramo u preostalih $N-1$ objekata uzeti $K-1$ istih, a ako ga nećemo uzeti, onda u preostalim objektima moramo uzeti njih K, pa prema tome možemo napisati relaciju:

$$\binom{N}{K} = \binom{N-1}{K-1} + \binom{N-1}{K}.$$

(Ova formula zove se **Pascalova formula**.[3]) Ovakav način rješavanja uvijek će izračunati točnu vrijednost broja pod uvjetom da krajnji rezultat stane u korišteni tip podataka:

[2] Po definiciji $M! = M(M-1)\cdots 2 \cdot 1$.
[3] **Blaise Pascal**, (1623. – 1662.) veliki francuski matematičar

$\pi \mathrm{l} \alpha y \sqrt[]{\mathbf{mat}\chi}$

funkcija $B(n,k)$
ako smo već izračunali vrijednost za (n,k) vrati tu vrijednost
ako $k=1$ onda vrati n
ako $n=k$ onda vrati 1
ako $k>n$ onda vrati 0
vrati i zapamti vrijednost $B(n-1,k-1)+B(n-1,k)$

✓

Uvijek treba izračunati sve vrijednosti koje možemo izravno dobiti.

Primjer 4. Koliko ima riječi sastavljenih od znakova {'a', 'b', 'c'} duljine $n \leq 30$ koje u sebi ne sadrže riječ S? Za $n=3$, $S=$"ab", program bi trebao vratiti vrijednost 21 (aaa, aac, aca, acb, acc, baa, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cac, cba, cbb, cbc, cca, ccb, ccc).

Rješenje. Koristili bismo ovakav algoritam da želimo prvo generirati sve takve riječi:

procedura generiraj(n, riječ R)
ako su je sufiks od R jednak S-u tada prekini izvršavanje
ako $n=0$ ispiši R te prekini izvršavanje
generiraj($n-1$, R+'a')
generiraj($n-1$, R+'b')
generiraj($n-1$, R+'c')

Svaki put kada pokušamo staviti zabranjenu riječ S, program će prekinuti daljnje grananje te riječi, ali kada bismo kao i ranije pokušali proceduru generiraj promijeniti u funkciju prebroji, ne bismo dobili nikakvo ubrzanje jer se na prvi pogled stanja uopće ne poklapaju (funkcija neće nikada biti pozvana više puta s istim argumentom). Potom možemo uvidjeti da ako je $S=$"bbb", da će prebroji(10, "aa") te prebroji(10, "bc") vratiti uvijek isti rezultat. Odgovor na pitanje *zašto je to tako* leži u činjenici da ovo što nas zanima nije cijela riječ koju smo prije generirali nego samo njezin najveći sufiks koji je ujedno i prefiks zabranjene riječi S. Imajući to na umu, jasno je da se stanje može opisati preostalom duljinom, te brojem slova u sufiksu generirane riječi koja podudaraju sa riječju S. Takav pristup dovodi nas do algoritma sa najvećom složenosti $30 \cdot 30$:

funkcija prebroji(n, riječ R)
odbaci prefiks od R koji ne može sudjelovati u daljnjem stvaranju riječi S
ako smo izračunali vrijednost za (n,R), vratimo ju
suma $=0$
suma = suma + prebroji($n-1$, R+'a')
suma = suma + prebroji($n-1$, R+'b')
suma = suma + prebroji($n-1$, R+'c')
vrati te zapamti suma

✓

Dakle, osnovna ideja svih ovakvih problema je da nađemo zajedničko stanje pomoću kojeg možemo opisati objekt koji moramo napraviti, a potom nađemo relaciju između stanja. Ponekad stanje nije toliko očito kao u primjeru 2. i 4., gdje smo morali uvesti nove argumente funkciji kako bismo efikasno mogli izračunati traženi podatak, ali u većini zadataka ono je vidljivo na prvi pogled, pogotovo nakon nekoliko provježbanih zadataka.

Zadatci za vježbu

Za kraj slijedi nekoliko zadataka za vježbu:

1. Na koliko načina možemo pravilno napisati $n \leq 50$ otvorenih/zatvorenih zagrada? Za n = 3 rezultat je 5, a nizovi su: "((()))", "()(())", "(())()", "(()())" te "()()()".
ULAZ: 7 REZULTAT: 1430

$\pi\mathrm{l}\alpha y\sqrt{\mathbf{mat}\chi}$

2. Koliko ima različitih permutacija brojeva $1, 2, \ldots N \leq 50$ takvih da oni tvore točno K rastućih podnizova (uzastopnih znakova)? Niz 1, 3, 2 ima dva rastuća podniza $(1, 3)$ te (2).

3. Na koliko načina možemo grupu od $N \leq 13$ ljudi jedinstvenih visina poredati u neki redoslijed tako da točno A njih vidi lijevo, a točno B njih desno? Osoba vidi na neku stranu ako joj se u tom smjeru ne nalazi nijedna osoba viša od nje.
ULAZ: 10 4 4 REZULTAT: 90720

ULAZ: 3 1 2 REZULTAT: 1

4. Nađi sumu $f(1) + f(2) + \ldots f(n)$ gdje $n \leq 10000$ ako definiramo:

$$f(x) := \begin{cases} 0, & \text{ako je } x = 1; \\ 1 + f(x/2), & \text{ako je } x \text{ paran}; \\ 1 + f(3x + 1), & \text{inače.} \end{cases}$$

ULAZ: 2 REZULTAT: 3

5. Koliko ima riječi duljine $n \leq 39$ sastavljenih od {'a', 'b', 'c'} takvih da u sebi ne sadrže ni S_1 ni S_2?

ULAZ: 3 ab ba REZULTAT: 17
ULAZ: 5 abc abb REZULTAT: 189

6. Na koliko načina možemo izabrati podskup od riječi S (S je duljine manja ili jednaka 50) takav da kada obrišemo sve znakove indeksi koji su sadržani u tom podskupu, dobijemo *palindrom*, tj. riječ koja se jednako čita sa lijeve i desne strane ("a", "aba", "autootua" su primjeri palindroma dok "mama" ili prazna riječ nisu palindromi)?

ULAZ: aaaa REZULTAT: 15
ULAZ: baobab REZULTAT: 22

Pojašnjenje 1. ulaza: kakav god podskup izabrali, osim kada bismo brisali sve znakove, rezultat je uvijek palindrom. Budući da postoji 2^4 podskupa od 4 znaka, a 1 nije dozvoljen, rješenje jest $2^4 - 1 = 15$.

Hint: proširite stanje još jednim argumentom da ne biste brojali iste podskupove više puta.