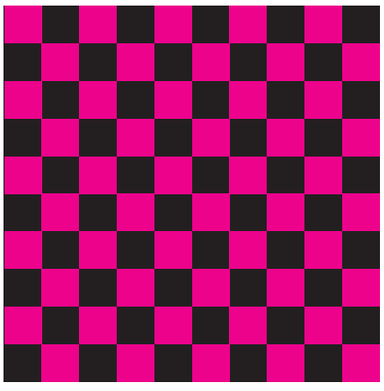
motywy Logo

Losowanie kwadratów

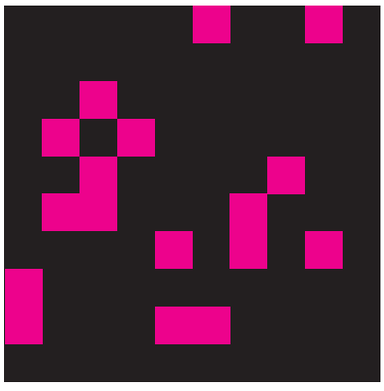
Andrzej WALAT

Wiele lat temu ułożyłem na Warszawski Konkurs Informatyczny dla uczniów szkół podstawowych (wówczas ośmioletnich) następujące

Zadanie. Rysunek 1 przedstawia regularną stupolową niebiesko-czarną szachownicę, a rysunek 2 nieregularną szachownicę o takich samych rozmiarach, na której 17 losowo wybranych pól jest niebieskich, a pozostałe są czarne. Napisz procedurę `szach :k`, która dla dowolnej nieujemnej liczby całkowitej $k \leqslant 100$ tworzy rysunek stupolowej szachownicy, której k losowo wybranych pól ma kolor niebieski, a pozostałe są czarne.



Rys. 1

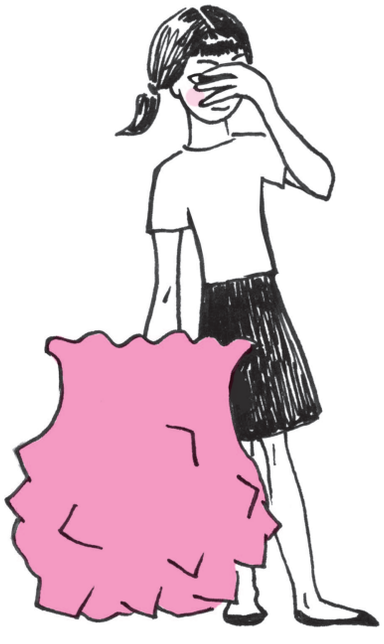


Rys. 2

W tym artykule omówię dwie propozycje rozwiązania zadania i przy okazji odniosę się do problemu trochę szerszego.

Rozwiązanie 1. Pola szachownicy numerujemy od 0 do 99. Głównym elementem rozwiązania jest następująca procedura.

```
oto szach :k
namalujCzarneTło
niech "lwp stoLiczbOdODo99
powtórz :k
[
  niech "nrPola los :lwp
  pomalujPole :nrPola
  przyp "lwp bezElementu :nrPola :lwp
]
już
```



Pierwsze polecenie powoduje narysowanie czarnego kwadratu przedstawiającego szachownicę, której wszystkie pola są czarne, ale trzeba je oczywiście zdefiniować. Następnie tworzymy zmienną roboczą `lwp` (lista wolnych pól), której wartością będzie lista 100 liczb naturalnych od 0 do 99. Można ją wpisać literalnie zamiast `stoLiczbOdODo99`, albo zdefiniować funkcję, której wynikiem będzie taka lista. Polecenie iteracyjne `powtórz` powoduje k-krotne wylosowanie numeru z listy, pomalowanie na niebiesko pola o wylosowanym numerze i usunięcie numeru pola z listy wolnych pól.

Rozwiązanie 2. Główna procedura rozwiązania jest w tym przypadku jeszcze prostsza. Po narysowaniu czarnego kwadratowego tła k razy malujemy na niebiesko pole o numerze wybranym losowo z zakresu od 0 do 99.

```
oto szach :k
namalujCzarneTło
powtórz :k [pomalujPole losowePole losowa 100]
już
```

Ale czy to jest na pewno dobry algorytm? Czy nie może się zdarzyć, że wylosujemy kilka razy ten sam numer pola i w rezultacie liczba pól niebieskich będzie mniejsza niż dana liczba k? Przed tym zabezpieczamy się, definiując

odpowiednio funkcję wyboru pola `losowePole`. Musi mieć ona daną liczbę n z zakresu od 0 do 99. Jeśli pole o danym numerze n jest czarne, to wynikiem jest n, a w przeciwnym przypadku – wynik kolejnego losowania.

```
oto losowePole :n
jeśli czarnePole? :n [wynik :n]
wynik losowePole losowa 100
już
```

Do kompletu potrzebne są jeszcze trzy procedury pomocnicze: `namalujTło` powoduje namalowanie czarnej szachownicy, `pomalujPole` powoduje pomalowanie na niebiesko pola o danym numerze, a funkcja `czarnePole?` sprawdza, czy pole o danym numerze jest czarne.

```
oto namalujTło
pod napoz [-200 -200]
ukp "czarny
wielokąt [4 [400 90]]
ukp "niebieski
już

oto pomalujPole :n
napoz [-200 -200] + 40 * zd div :n 10 mod :n 10
wielokąt [4 [40 90]]
już

oto czarnePole? :n
wy "czarny = s1' kolPktT [-180 -180] + 40 * zd div :n 10 mod :n 10
już
```

Komentarz. Oba rozwiązania są losowe, ponieważ wynik jest przypadkowy, ale drugie jest losowe podwójnie. W pierwszym przypadku liczba operacji prowadzących do wyniku jest jednoznacznie wyznaczona przez daną k. Zawsze k razy wykonujemy trzy operacje: wybranie losowego numeru z aktualnej listy wolnych pól, pomalowanie na niebiesko pola o wybranym numerze oraz usunięcie wybranego numeru z listy.

W drugim przypadku k razy losujemy numer zawsze z zakresu od 0 do 99 i sprawdzamy: jeśli pole o wybranym numerze jest niebieskie, to powtarzamy losowania tak długo, aż wypadnie numer pola, które jest czarne. W rezultacie łączna liczba losowań jest zawsze nie mniejsza niż k, ale może być znacznie większa. Czy wybrana strategia nie jest zbyt ryzykowna? Czy gdy k jest duże (większe niż 50), nie może się zdarzyć, że po wylosowaniu i zamalowaniu na niebiesko 50 pól będziemy następnie losowali w kółko numery tych samych pól i czekali bez skutku na zakończenie zadania? Warto zweryfikować pomysł, eksperymentalnie i teoretycznie. Po napisaniu trzech poleceń: `pisz czas szach 100 pisz czas` komputer powinien pomalować wszystkie pola na niebiesko i wypisać czas przed rozpoczęciem i po wykonaniu zadania. Na moim laptopie zajęło to niecałe 0,2 sekundy. Wielokrotne powtórzenie eksperymentu dało ten sam wynik. Rozwiązanie 2 można jeszcze ulepszyć, stosując sprytny chwyt. Zadanie pomalowania na niebiesko k pól czarnej stupolowej szachownicy jest równoważne z zadaniem pomalowania na czarno $100-k$ pól niebieskiej szachownicy. Zadanie możemy zacząć od sprawdzenia, czy dana liczba pól do pomalowania jest większa niż połowa liczby wszystkich pól i jeśli tak, najpierw pomalować całą szachownicę na niebiesko, a następnie $n = 100-k$ losowo wybranych pól na czarno.

Z teoretycznego punktu widzenia ocena efektywności rozwiązania 2 sprowadza się do odpowiedzi na pytanie: Ile średnio losowań liczby z zakresu od 0 do $n-1$ trzeba wykonać, by otrzymać k różnych wyników? Odpowiedzią jest stosunkowo prosty wzór (dowód w aneksie na stronie WWW):

$$T(k,n) = n\left(\frac{1}{n} + \frac{1}{n-1} + \ldots + \frac{1}{n-k+1}\right)$$

W przypadku, gdy $k \leqslant n/2$ średnia liczba losowań jest istotnie mniejsza niż $2k$. Porównując dwa rozwiązania, musimy wziąć pod uwagę jeszcze jeden fakt. Nie wszystkie operacje mają taką samą wagę. Na poziomie języka Logo operacja usunięcia elementu z listy jest elementarna, jest to tylko jeden krok obliczeń. Faktycznie jest to jednak operacja złożona. Istnieje wiele sposobów reprezentowania list i realizacji operacji na listach. Nie sposób ich tu wszystkich opisać. Jeśli na przykład elementy listy zajmują kolejne komórki w pamięci, to usunięcie jednego elementu powoduje konieczność przesunięcia wszystkich kolejnych o jedno miejsce. Nie jest to operacja jednokrokowa i czas jej wykonania jest proporcjonalny do długości listy.

Ambitnym Czytelnikom proponuję modyfikację przedstawionych procedur, by można je było stosować do szachownic o większych rozmiarach (na przykład 100×100) i praktyczne porównanie efektywności obu rozwiązań.