Programowanie

Dariusz Wardecki, wyk. II

Powtórzenie

Co wypisze program?

```
...
char x, y, z;
x = '1';
y = '3';
z = x + y;
cout << z << endl;
...
```

Powtórzenie

Znajdź błąd w poniższym fragmencie kodu?

```
…
short int x, y;
x = 65535;
y = x + 1;

cout << y << endl;
…
```

Algorytmy

Algorytm

Skończony zbiór dobrze zdefiniowanych instrukcji przeznaczony do wykonania określonego zadania, który przy ustalonym stanie początkowym pozwala na uzyskanie odpowiedniego, rozpoznawalnego stanu końcowego w skończonym czasie.

W uproszczeniu

Metoda poszukiwania (lub tworzenia) rozwiązania zadanego problemu.

Algorytmy

Stan początkowy dla algorytmu

Dane wejściowe (*ang. input data*).

Stan końcowy dla algorytmu

Wynik (*ang. result*).

Definicja poprawności algorytmu

Algorytm jest poprawny (*ang. correct*), gdy dla każdych dopuszczalnych danych wejściowych jednocześnie spełnione są dwa następujące warunki:

1. Wynik jest otrzymywany w skończonej liczbie kroków — problem zatrzymania (stopu).
2. Wynik stanowi rozwiązanie problemu, dla którego algorytm został stworzony.

Algorytmy

Przykłady:

- Oblicznie reszty z dzielenia (operacja modulo)
- Algorytm Euklidesa
- Sito Eratostenesa
- Uniwersalny algorytm mnożenia (russian peasant algorithm)
- Przeszukiwanie binarne (*ang. binary search*) lub bisekcja (*ang. bisection*)

Algorytmy

Obliczanie reszty z dzielenia
niech n i m będą liczbami naturalnymi, n > m

1. Niech k = n
2. Jeśli k < m, to k jest resztą z dzielenia, k = n%m
3. Niech k = k - m
4. Przejdź do kroku 2

Algorytmy

Algorytm Euklidesa

Niech n i m będą liczbami nauralnymi, n > m. Wyznaczamy największy wspólny dzielnik n i m, czyli NWD(n, m)

Obserwacja

Jeśli r jest resztą z dzielenia n przez m, to n = km + r (dla pewnego całkowitego k), więc NWD(n, m) = NWD(m, r)

Algorytmy

Algorytm Euklidesa

Niech a, b i r będą miejscami do przechowywania wyników

1. Niech a = n i b = m.
2. Niech r = a%b (resztę z dzielenia a/b zapisujemy w r).
3. Jeśli r = 0, to NWD(n, m) = b.
4. Niech a = b i b = r.
5. Przejdź do kroku 2.

Algorytmy

Sito Eratostenesa - chcemy ustalić, które z liczb naturalnych nie większych od zadanego N są liczbami pierwszymi.

A = {2,3, …, N}, k i m - miejsca do przechowywania pośrednich wyników

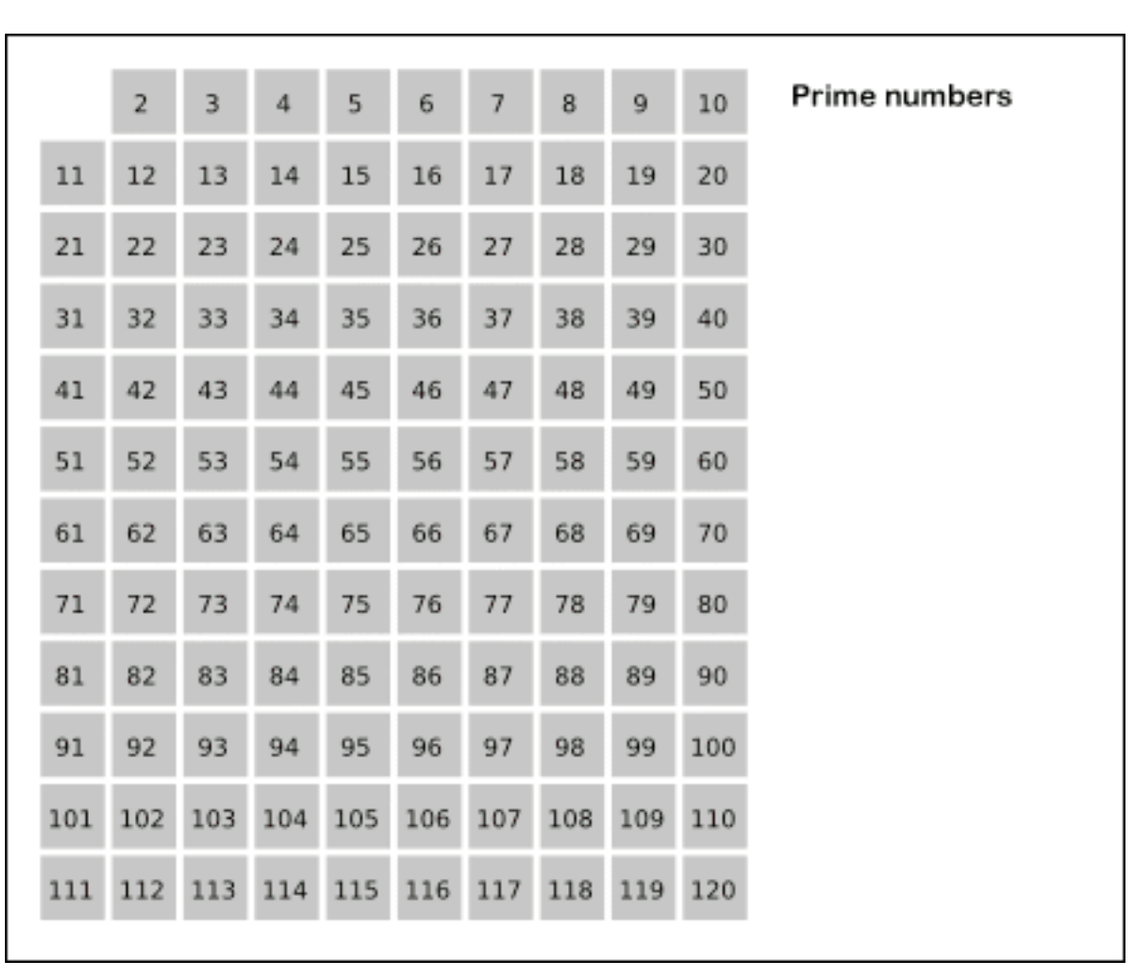
1) Niech k = 2
2) Jeśli k^2 > N, zbiór A zawiera tylko liczby pierwsze
3) Usuwamy z A wszystkie wielokrotności k, począwszy od k^2
4) W m zapisz najmniejszą liczbę ze zbioru A większą od k.
5) Niech k = m.
6) Przejdź do kroku 2).

Algorytmy

Sito Eratostenesa - chcemy ustalić, które z liczb naturalnych nie większych od zadanego N są liczbami pierwszymi.

A = {2,3, …, N}, k i m - miejsca do przechowywania pośrednich wyników

1) Niech k = 2
2) Jeśli k^2 > N, zbiór A zawiera tylko liczby pierwsze
3) Usuwamy z A wszystkie wielokrotności k, począwszy od k^2
4) W m zapisz najmniejszą liczbę ze zbioru A większą od k.
5) Niech k = m.
6) Przejdź do kroku 2).



Algorytmy

Uniwersalny algorytm mnożenia

Russian peasant algorithm

Obliczamy wynik mnożenia dwóch nieujemnych liczb całkowitych a i b:

1. Niech $a_0 = a$, $b_0 = b$ i $k = 0$.
2. Niech $k = k + 1$.
3. Niech $a_k = a_{k-1}/2$ (dzielenie bez reszty) i $b_k = 2b_{k-1}$.
4. Jeśli $a_k > 1$, to przejdź do kroku 2.
5. Wynik mnożenia jest sumą wszystkich b_k ($k = 0, 1, 2, \ldots$), dla których odpowiadające im a_k są nieparzyste.

Algorytmy

Uniwersalny algorytm mnożenia

Obliczamy iloczyn 156×18

1. Wyznaczamy ciągi a_k i b_k:

k	a_k	b_k
0	156	18
1	78	36
2	39	72
3	19	144
4	9	288
5	4	576
6	2	1152
7	1	2304

2. $156 \times 18 = 2304 + 288 + 144 + 72 = 2808$

Algorytmy

Uniwersalny algorytm mnożenia

Obliczamy iloczyn 156×18

1. Można sumować na bieżącc

k	a_k	b_k	s_k
0	156	18	0
1	78	36	0
2	39	72	72
3	19	144	216
4	9	288	504
5	4	576	504
6	2	1152	504
7	1	2304	2808

2. $156 \times 18 = s_7 = 2808$

Algorytmy

Uniwersalny algorytm mnożenia

Obliczamy iloczyn dwóch nieujemnych liczb całkowitych a i b

Niech A, B i S będą miejscami służącymi do przechowywania pośrednich wyników.

1. Niech $A = a$, $B = b$ i $S = 0$.
2. Jeśli $A \,\%\, 2 = 1$ (wartość w A jest nieparzysta), to $S = S + B$.
3. Jeśli $A = 1$, przejdź do kroku 6.
4. Niech $A = A/2$ (dzielenie bez reszty) i $B = 2B$.
5. Przejdź do kroku 2.
6. Wynik mnożenia jest zapisany w S.

Algorytmy

Przeszukiwanie binarne (bisekcja)

Poszukiwanie elementu r w skończonym i uporządkowanym zbiorze B

1. Niech $B_0 = B$ i $k = 0$.
2. Jeśli wszystkie elementy zbioru B_k są jednakowe lub B_k jest zbiorem pustym, przejdź do kroku 6.
3. Niech m_k oznacza medianę zbioru B_k.
4. Jeśli $r \leqslant m_k$, to niech $B_{k+1} = \{b \in B_k : b \leqslant m_k\}$, a w przeciwnym wypadku niech $B_{k+1} = \{b \in B_k : b > m_k\}$.
5. Niech $k = k + 1$ i przejdź do kroku 2.
6. Jeśli $r \in B_k$, to $r \in B$.

Algorytmy

Przeszukiwanie binarne (bisekcja) - przykład

Weźmy $B = \{3, 7, 21, 48, 56, 122, 141, 218, 225, 248\}$ i $r = 225$

k	B_k	m_k
0	$\{3, 7, 21, 48, 56, 122, 141, 218, 225, 248\}$	56
1	$\{122, 141, 218, 225, 248\}$	218
2	$\{225, 248\}$	225
3	$\{225\}$	225

Złożoność obliczeniowa algorytmu

Złożoność obliczeniowa algorytmu

Złożoność obliczeniowa algorytmu

Miara liczby operacji przeprowadzanych podczas wykonywania programu stanowiącego realizację (zapis) danego algorytmu.

Złożoność obliczeniowa algorytmu

Złożoność obliczeniowa algorytmu

Miara liczby operacji przeprowadzanych podczas wykonywania programu stanowiącego realizację (zapis) danego algorytmu.

Obserwacje

1. Złożoność obliczeniowa algorytmu jest funkcją rozmiarów jego danych wejściowych, czyli rozmiaru zadania.
2. Zwykle jest ona szacowana (w przybliżeniu) dla rozmiaru zadania dążącego do nieskończoności.
3. Czas przeprowadzania obliczeń jest ściśle zależny od złożoności obliczeniowej algorytmu.

Złożoność obliczeniowa algorytmu

Dla rozmiaru zadania reprezentowanego przez jedną liczbę N

Stała – liczba operacji nie zależy od N.

Logarytmiczna – liczba operacji jest proporcjonalna do $\log N$.

Liniowa – liczba operacji jest wprost proporcjonalna do N.

Wielomianowa – liczba operacji jest proporcjonalna N^k, gdzie $k > 1$.

Złożoność obliczeniowa algorytmu

Dla rozmiaru zadania reprezentowanego przez jedną liczbę N

Stała – liczba operacji nie zależy od N.

Logarytmiczna – liczba operacji jest proporcjonalna do $\log N$.

Liniowa – liczba operacji jest wprost proporcjonalna do N.

Wielomianowa – liczba operacji jest proporcjonalna N^k, gdzie $k > 1$.

Często czas działania programu wyraża się poprzez jego złożoność obliczeniową, także na przykład jeśli algorytm ma wielomianową złożoność obliczeniową mówi się, że jest on wykonywany „w czasie wielomianowym”.

Złożoność obliczeniowa

Przykłady:

Złożoność obliczeniowa

Przykłady:

Dla wyznaczania reszty z dzielenia

Liniowa względem ilorazu n/m.

Złożoność obliczeniowa

Algorytm Euklidesa:

W najgorszym wypadku logarytmiczna względem n (zakładając, że obliczanie reszty z dzielenia ma stałą złożoność).

1. W najgorszym wypadku w każdym kroku mamy

$$b \sim r \sim \frac{a}{2}$$

Złożoność obliczeniowa

Algorytm Euklidesa:

W najgorszym wypadku logarytmiczna względem n (zakładając, że obliczanie reszty z dzielenia ma stałą złożoność).

1. W najgorszym wypadku w każdym kroku mamy

$$b \sim r \sim \frac{a}{2}$$

2. Zatem (w najgorszym wypadku) w kroku k

$$b \sim r \sim \frac{n}{2^k}$$

Złożoność obliczeniowa

Algorytm Euklidesa:

W najgorszym wypadku logarytmiczna względem n (zakładając, że obliczanie reszty z dzielenia ma stałą złożoność).

1. W najgorszym wypadku w każdym kroku mamy

$$b \sim r \sim \frac{a}{2}$$

2. Zatem (w najgorszym wypadku) w kroku k

$$b \sim r \sim \frac{n}{2^k}$$

3. Zatem (w najgorszym wypadku) w ostatnim kroku, czyli dla $k = s$, gdzie s jest całkowitą liczbą wykonanych kroków

$$1 \sim r \sim \frac{n}{2^s} \Longrightarrow s \sim \log_2 n$$

Złożoność obliczeniowa

Sito Eratostenesa:

Kwadratowa względem N.

1. Liczba powtórzeń „usuwania wielokrotności” jest rzędu N.

Złożoność obliczeniowa

Sito Eratostenesa:

Kwadratowa względem N.

1. Liczba powtórzeń „usuwania wielokrotności” jest rzędu N.
2. W każdym powtórzeniu przeprowadzamy $(N - k^2)/k$ operacji.

Złożoność obliczeniowa

Sito Eratostenesa:

Kwadratowa względem N.

1. Liczba powtórzeń „usuwania wielokrotności” jest rzędu N.
2. W każdym powtórzeniu przeprowadzamy $(N - k^2)/k$ operacji.
3. Łączna liczba przeprowadzanych operacji s jest proporcjonalna do

$$\frac{N\left(N - \langle k \rangle^2\right)}{\langle k \rangle} = \frac{N^2}{\langle k \rangle} - \langle k \rangle N$$

gdzie $\langle k \rangle$ jest średnią wartością k.

Złożoność obliczeniowa

Sito Eratostenesa:

Kwadratowa względem N.

1. Liczba powtórzeń „usuwania wielokrotności” jest rzędu N.
2. W każdym powtórzeniu przeprowadzamy $(N - k^2)/k$ operacji.
3. Łączna liczba przeprowadzanych operacji s jest proporcjonalna do

$$\frac{N\left(N - \langle k \rangle^2\right)}{\langle k \rangle} = \frac{N^2}{\langle k \rangle} - \langle k \rangle N$$

   gdzie $\langle k \rangle$ jest średnią wartością k.
4. Przy N dążącym do nieskończoności pierwszy człon jest dominujący, więc mamy

$$s \sim N^2$$

Złożoność obliczeniowa

Mnożenie uniwersalne:

Logarytmiczna względem *a* przy założeniu, że dodawanie ma stałą złożoność.

1. Mamy $a_k = a/2^k$.

Złożoność obliczeniowa

Mnożenie uniwersalne:

Logarytmiczna względem *a* przy założeniu, że dodawanie ma stałą złożoność.

1. Mamy $a_k = a/2^k$.
2. Zatem, dla k równego liczbie kroków s dostajemy $1 = a_s = a/2^s$.

Złożoność obliczeniowa

Mnożenie uniwersalne:

Logarytmiczna względem *a* przy założeniu, że dodawanie ma stałą złożoność.

1. Mamy $a_k = a/2^k$.
2. Zatem, dla k równego liczbie kroków s dostajemy $1 = a_s = a/2^s$.
3. Stąd wynika, że $s \sim \log_2 a$.

Złożoność obliczeniowa

Bisekcja:

Logarytmiczna względem liczby elementów B.

1. Dla każdego k moc (liczba elementów) zbioru B_{k+1} jest średnio o połowę mniejsza od mocy (liczby elementów) zbioru B_k.

Złożoność obliczeniowa

Bisekcja:

Logarytmiczna względem liczby elementów B.

1. Dla każdego k moc (liczba elementów) zbioru B_{k+1} jest średnio o połowę mniejsza od mocy (liczby elementów) zbioru B_k.

2. Zatem dla $k = s$ (liczba kroków) moc zbioru B_s jest rzędu mocy zbioru B podzielonej przez 2^s.

Złożoność obliczeniowa

Bisekcja:

Logarytmiczna względem liczby elementów B.

1. Dla każdego k moc (liczba elementów) zbioru B_{k+1} jest średnio o połowę mniejsza od mocy (liczby elementów) zbioru B_k.

2. Zatem dla $k = s$ (liczba kroków) moc zbioru B_s jest rzędu mocy zbioru B podzielonej przez 2^s.

3. Stąd wynika, że $s \sim \log_2 \bar{B}$.

Równoważność algorytmów

Równoważność algorytmów

Definicja równoważności algorytmów

Dwa algorytmy można uznać za równoważne (*ang. equivalent*), gdy:

1. Każdy z nich pozwala wyznaczyć rozwiązanie tego samego problemu.
2. Mają one jednakową złożoność obliczeniową (z dokładnością do stałego czynnika).