Malachoviacus Informaticus

Etap II, 12 kwietnia 2014 r.

Instrukcje dla zawodników

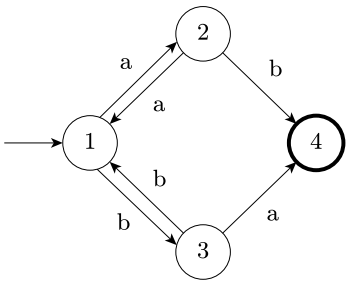
Arkusze otwieramy na wyraźne polecenie komisji. Wszystkie poniższe instrukcje zostaną odczytane i dokładnie wyjaśnione.

1. Arkusz składa się z *3 zadań*.
2. Każde zadanie składa się z *wprowadzenia* oraz *trzech* pytań.
3. Pytania w każdym zadaniu warte są kolejno 3, 6 i 9 punktów. Za każde zadanie można więc uzyskać maksymalnie 18 punktów.
4. Przed udzieleniem odpowiedzi na pytania przeczytaj dokładnie wprowadzenie oraz treści poleceń.
5. Swoje odpowiedzi na pytania zapisz *czytelnie* na przeznaczonych do tego arkuszach.
6. Do zapisu odpowiedzi używaj wyłącznie *długopisu lub pióra* z *czarnym lub niebieskim* tuszem. Do wykonywania rysunków możesz użyć ołówka.
7. Każdy arkusz powinien zawierać odpowiedź (lub jej część) na *tylko jedno* zadanie.
8. Przed rozpoczęciem rozwiązywania zadań na kartce z kodem zapisz wyraźnie swoje imię i nazwisko.
9. W prawym górnym rogu każdego arkusza odpowiedzi zapisz czytelnie *swój kod* oraz *numer zadania*.
10. Użycie kalkulatora jest dozwolone, jednak żadne z zadań tego nie wymaga.
11. Czas na rozwiązanie zadań wynosi *180 minut*.

Zadanie 1

Automaty skończone

Spójrz na automat skończony na diagramie poniżej.



Jeżeli, zaczynając od stanu początkowego, będziemy wykonywać przejścia zgodnie z łańcuchem znaków *aaba*, to otrzymamy pewien ciąg stanów oraz przejść między nimi. Możemy ten ciąg zapisać następująco:

$$1 \xrightarrow{a} 2 \xrightarrow{a} 1 \xrightarrow{b} 3 \xrightarrow{a} 4$$

Z kolei łańcuch *bbaa* wygeneruje następujący ciąg stanów i przejść:

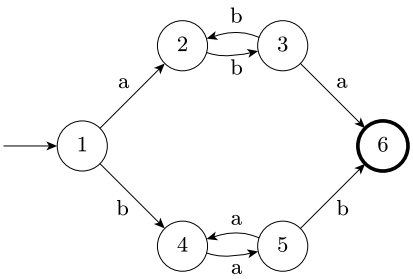
$$1 \xrightarrow{b} 3 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{a} 1$$

Wyjście poza automat możemy oznaczyć poprzez $\varnothing$, na przykład dla ciągu znaków *abbb* (zwróć uwagę, że przejść jest mniej niż znaków w łańcuchu):

$$1 \xrightarrow{a} 2 \xrightarrow{b} 4 \xrightarrow{b} \varnothing$$

Pytania

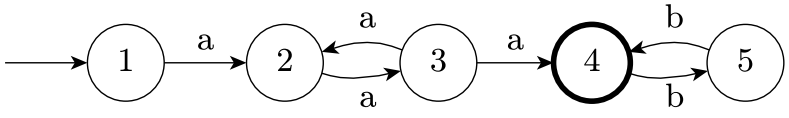
Pytanie 1 Spójrz na automat poniżej.



Które z następujących łańcuchów zostaną zaakceptowane przez ten automat: *ab*, *bab*, *bbaa*, *abbba*, *babba*, *baca*?

Zapisz ciągi stanów i przejść, tak jak w opisie powyżej, które zostaną wygenerowane przez te łańcuchy w danym automacie.

PYTANIE 2 Spójrz na automat poniżej.



Podaj 3 przykłady łańcuchów, które ten automat zaakceptuje oraz 3 przykłady, które odrzuci. Podaj odpowiednie ciągi stanów i przejść. Opisz słownie (jak najprecyzyjniej) wszystkie łańcuchy znaków, które ten automat akceptuje.

PYTANIE 3 Narysuj automat, który będzie akceptował łańcuchy składające się wyłącznie z parzystej liczby liter a, np. aa, $aaaa$ itd.

Następnie narysuj automat, który będzie akceptował łańcuchy składające się z liter a i b, jednak tylko takie, w których liczba liter a jest parzysta, np. aa, $abba$, $babbabaa$ itp.

Na końcu narysuj automat, który będzie akceptował łańcuchy składające się z liter a i b, jednak tylko takie, w których liczba liter a jest parzysta, a liczba liter b jest nieparzysta, np. aab, bbb, $bababaa$ itp.

Upewnij się, że stany początkowe i akceptujące są wyraźnie zaznaczone. Możesz użyć innych oznaczeń, jednak w takim przypadku wyraźnie zaznacz jakie oznaczenia przyjąłeś.

Uwaga! Żaden z powyższych automatów nie powinien mieć więcej niż 5 różnych stanów.

Zadanie 2

Maszyna stosowa

Dana jest maszyna. Podłączona jest ona do monitora i klawiatury, a także do jednostki pamięci działającej jak stos. Stos początkowo jest pusty.

Maszyna ta przyjmuje programy, które składają się z ciągów poleceń. Polecenia rozumiane przez maszynę to:

1. `push x` – umieszcza wartość x na szczycie stosu
2. `pop` – usuwa wartość ze szczytu stosu
3. `read` – prosi użytkownika o wpisanie liczby i umieszcza wprowadzoną wartość na szczycie stosu
4. `write` – wypisuje na ekran wartość ze szczytu stosu
5. `add` – zdejmuje dwie wartości ze szczytu stosu, oblicza ich sumę i umieszcza wynik na szczycie stosu
6. `mul` – zdejmuje dwie wartości ze szczytu stosu, oblicza ich iloczyn i umieszcza wynik na szczycie stosu
7. `dup` – zdejmuje wartość ze szczytu stosu i umieszcza ją ponownie na szczycie, ale dwukrotnie

Maszyna wykonuje polecenia po kolei. Stan stosu przed wykonaniem polecenia jest taki sam jak po wykonaniu poprzedniego polecenia (to znaczy stan stosu zapisuje się i nikt inny nie ma do niego dostępu w czasie działania programu). Jeżeli maszyna będzie musiała zdjąć ze stosu więcej elementów niż się na nim znajduje, zwrócony zostanie błąd i działanie programu zostanie przerwane.

PYTANIA

PYTANIE 1 W pewnym momencie działania programu stos zawiera wartości (od szczytu): 5, 3, 2. Jak wyglądałby stos po wykonaniu następnego polecenia, jeżeli tym poleceniem byłoby:

1. `add`
2. `dup`
3. `write`

PYTANIE 2 Do maszyny przekazano program:

```
push 2
push 5
add
push 3
mul
write
```

Jak będzie wyglądał stos po wykonaniu każdego z poleceń? Jaka wartość zostanie wypisana na ekran? Zapisz działanie, którego wynik oblicza ten program.

PYTANIE 3 Napisz program, który pobierze od użytkownika 3 liczby i wypisze na ekran sumę ich kwadratów. Na przykład dla liczb 3, 4 i 5 wynikiem powinno być 50.

Podpowiedź: pomiędzy proszeniem użytkownika o wprowadzanie kolejnych liczb możesz wykonywać inne operacje.

Zadanie 3

Obliczenia równoległe po raz drugi

W pierwszym etapie przyjrzeliśmy się problemom związanym ze współdzieleniem zasobów przez kilka procesorów. Tym razem zobaczymy, że zwiększanie liczby procesorów nie zawsze ma sens.

Intuicyjnie może wydawać się, że jeżeli użyjemy dwóch procesorów, to całość będzie działać jak jeden, dwa razy szybszy procesor. Nie jest to jednak prawdą.

Dla lepszego zrozumienia znów posłużymy się analogią. Pewna firma przewozowa dostała zlecenie przewiezienia 8 osób z miejsca A do miejsca B, a następnie powrotu do miejsca A. Jeden samochód może zabrać maksymalnie 4 pasażerów, a odległość z punktu A do B przebywa w 10 minut (20 minut w obie strony).

Gdyby do wykonania tego zlecenia użyć jednego samochodu, całość zajmie 40 minut (2 kursy po 4 osoby, każdy kurs trwa 20 minut). Gdybyśmy zaś użyli dwóch samochodów, to całość zajmie tylko 20 minut, a więc faktycznie dwa razy krócej.

Co jednak, gdyby do przewiezienia były tylko 4 osoby? Jeden samochód wykona zlecenie w 20 minut, dwa samochody również będą potrzebować do tego 20 minut.

Zatem dwa samochody (procesory) nie wykonają pracy dwa razy szybciej, jednak mogą jej w tym samym czasie wykonać dwa razy *więcej*. Jeżeli nie ma wystarczająco dużo pracy do podziału, dodatkowe procesory są zbędne.

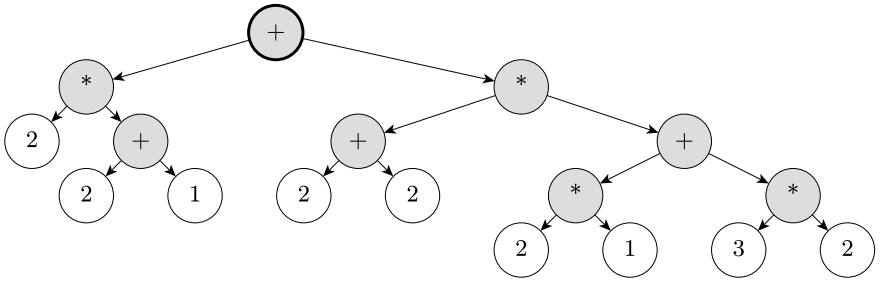
Przy ocenie słuszności dodawania kolejnych procesorów możemy posłużyć się czasem T_n. Zdefiniowany jest on jako ilość czasu potrzebna do wykonania danego zadania przez n procesorów (przy założeniu optymalnego podziału pracy). W naszym przykładzie z 8 pasażerami czas T_1 wynosi 40 minut, a czas T_2 20 minut.

Czas T_n będzie malał wraz ze wzrostem n, jednak tylko do pewnego momentu. Od pewnego n każdy kolejny czas będzie taki sam jak poprzedni. Ten czas będziemy oznaczać przez T_∞.

Ostatnim istotnym oznaczeniem będzie n_{max}. Jest to najmniejsze takie n, dla którego $T_n = T_\infty$. W naszym przykładzie z 8 pasażerami n_{max} wynosi 2, ponieważ $T_1 < T_2 = T_3 = T_\infty$.

Teraz musimy jeszcze znaleźć łatwy sposób na znalezienie wielkości T_1, T_∞ i n_{max} dla dowolnego obliczenia. W tym zadaniu, dla ułatwienia, ograniczymy się do prostych działań matematycznych.

Dla przykładu spójrzmy na działanie $2\cdot(2+1)+(2+2)\cdot(2\cdot1+3\cdot2)$. Przedstawmy je w postaci drzewa składniowego.



Przyjmijmy, że każda operacja zajmuje dokładnie 1 jednostkę czasu, nieza-

leżnie od rodzaju operacji i argumentów. Jeżeli posiadamy tylko jeden procesor, to wszystkie operacje muszą być wykonywane jedna po drugiej, więc ostatecznie będziemy potrzebować tyle czasu ile jest operatorów w drzewie. W naszym przykładzie więc $T_1 = 8$.

Czas T_∞ będzie równy *głębokości* drzewa. Głębokość definiujemy jako największą odległość operatora od korzenia (oznaczonego pogrubioną linią). *Odległość* z kolei to liczba strzałek, która dzieli korzeń od danego operatora. Głębokość naszego drzewa wynosi 3.

Z kolei n_{max} będzie równy *szerokości* drzewa. Tę wielkość definiujemy jako największą liczbę operatorów znajdujących się w jednakowej odległości od korzenia. W naszym drzewie mamy 2 wierzchołki w odległości 1, 3 wierzchołki w odległości 2 i 2 wierzchołki w odległości 3. Stąd szerokość drzewa będzie równa 3.

PYTANIA

PYTANIE 1 Firma przewozowa zakupiła pewną liczbę mikrobusów, z których każdy mieści 8 pasażerów. Otrzymała zlecenie przewiezienia 26 osób z punktu A do B. Przejazd między tymi punktami zajmuje 15 minut. Znajdź T_1, T_∞ i n_{max} tego zlecenia, jeżeli użyte zostały mikrobusy.

PYTANIE 2 Dane jest działanie $(2x + 3y) \cdot (6 + 3z)$, gdzie x, y i z są wartościami, które trzeba wczytać z pamięci. Wykonanie każdego działania matematycznego zajmuje jedną jednostkę czasu. Wczytanie jednej wartości z pamięci również zajmuje 1 jednostkę. Jeden procesor może wczytywać tylko jedną wartość w danym momencie, ale kilka procesorów może wczytywać dane jednocześnie.

Narysuj drzewo składniowe tego działania. Znajdź T_1, T_∞ i n_{max}. Narysuj kolejne etapy obliczeń przy użyciu n_{max} procesorów.

Podpowiedź: potraktuj wczytywanie wartości z pamięci jako operację arytmetyczną, która przyjmuje jeden argument.

PYTANIE 3 Silnia z n (zapisywane $n!$) to iloczyn liczb naturalnych od 1 do n włącznie. Na przykład $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.

Narysuj drzewo składniowe (składające się wyłącznie z liczb i operatora mnożenia) dla obliczenia 8!. Znajdź T_1, T_∞ i n_{max}. Istnieje więcej niż jedno możliwe drzewo. Postaraj się narysować takie, które posiada jak najniższą wartość T_∞.

Jeżeli potrzebujesz, możesz użyć zapisu $x \cdots y$ dla oznaczenia iloczynu liczb od x do y włącznie (tj. $2 \cdots 4 = 2{\cdot}3{\cdot}4$, $1 \cdots n = n!$, $n \cdots n = n$ itp.). Pamiętaj jednak, że rysowanie wszystkich stanów pośrednich drzewa składniowego nie jest wymagane.