Laboratorium algorytmów i struktur danych

Laboratorium 1

Zapoznanie z programem laboratorium, sposobem oceny ćwiczeń, szkolenie BHP. Zapoznanie ze środowiskiem programistycznym.

Zasady zaliczenia.

1. Na zajęciach oceniane są zmiany w programach oraz sam kod przygotowany wcześniej. Treść zmian jest podawana na zajęciach indywidualnie dla grupy.
2. Listy zadań są zadawane z wyprzedzeniem. Na każde kolejne zajęcia należy przyjść z samodzielnie przygotowaną listą.
3. Skala ocen z zajęć jest od 0 do 5,5. Ocena końcowa to średnia arytmetyczna z 14 list. Nieoddanie listy jest równoznaczne z 0 punktów za nią.
4. Żeby móc dostać ocenę pozytywną należy oddać powyżej $\frac{3}{4}$ list spośród opublikowanych przez prowadzącego.
5. Ocenę z listy można poprawiać w ciągu 10 dni roboczych od zajęć na których obowiązuje.
6. Na zajęciach dopuszczalne są 2 nieusprawiedliwione nieobecności które nie zwalniają od oddania listy. Każda kolejna nieobecność wymaga usprawiedliwienia lekarskiego, lub (w szczególnych przypadkach) od pracownika uczelni lub władz organizacji studenckiej[1].
7. Listy można odrabiać w innych grupach u tego samego prowadzącego, pod warunkiem, że jest miejsce w sali i jest to wcześniej uzgodnione.

Przydatne informacje.

Adres dokumentacji języka Java:
wersja 7: `http://docs.oracle.com/javase/7/docs/api/`
wersja 8: `http://docs.oracle.com/javase/8/docs/api/`

[1] Dotyczy jedynie działalności na rzecz uczelni wykraczającej po za zwykły tok studiów i jest dopuszczalna jednorazowo.

Laboratorium algorytmów i struktur danych

Laboratorium 2

Lista „rozgrzewkowa"

Zadanie 1. Kraina kawowa kofeiną płynąca rozwija się coraz szybciej. Od lat postępuje rozwój techniki cyfrowej, a lokalna uczelnia kształci wyrafinowanych specjalistów. Władca krainy, Kawuś VII, dba o nastroje wśród podwładnych, jak również o ich poziom intelektualny. Dlatego postanowił zorganizować konkurs na sprawdzanie własności liczb naturalnych. Ogłoszono dwie konkurencje: na parzystość i pierwszość[2] liczb. Zwrócono się do lokalnych ekspertów w celu automatyzacji testów.

Napisz program który:

1. Otwiera plik tekstowy z liczbami naturalnymi (1 linia, 1 liczba).
2. Wczytuje po kolei wszystkie liczby z pliku i określa, czy są: parzyste/nieparzyste oraz pierwsze/złożone.
3. Wykorzystuje enumeracje w wewnętrznej implementacji.
4. Wypisuje wyniki na ekran.

Zadanie 2. Po rozegraniu pierwszej rundy konkursu okazało się, że nośniki danych z plikami zawierającymi zadania konkursowe nie są idealne i wkradają się przekłamania. Postanowiono wprowadzić bity parzystości w celu weryfikacyjnym.

Napisz 2 programy, z których pierwszy:

1. Wczytuje plik z liczbami tak, jak w pierwszym zadaniu.
2. Wylicza bit parzystości[3] każdej liczby.
3. Zapisze do drugiego pliku wyniki w formacie 1 wierz - 1 suma.

Oraz drugi:

1. Wczytuje oba plik z liczbami i bitami tak, jak w pierwszym zadaniu.
2. Wylicza bit parzystości każdej liczby.
3. Sprawdza, czy bity się zgadzają i komunikuje na ekranie o tym.

Zadanie 3. (na 5,5) Okazało się, że sposób sprawdzania plików działa rewelacyjnie i postanowiono przepisać program tak, żeby dział ze wszystkimi plikami - również binarnymi.

Napisz program który:

1. Otworzy dowolny plik (np. obrazek) i policzy dla każdego bajtu bit parzystości.
2. Zapisze 1 bajt z bitami parzystości dla każdych 8 bajtów (z wyjątkiem końca pliku).
3. Pozwoli na weryfikację wyników.

[2]Liczba pierwsza, to taka która dzieli się jedynie przez 1 i samą siebie, np. 2, 3, 5, 7, 11, 13...

[3]W celu obliczenia bitu parzystości, należy dodać wszystkie bity w liczbie - wynikiem jest 1 dla nieparzystej sumy i 0 dla parzystej.

Laboratorium algorytmów i struktur danych

Laboratorium 3

Iteratory

Zadanie 1. Zabawa liczbami trwa w najlepsze. Całość rozwija się na tyle, żę władca postanowił ogłosić kolejną edycję konkursu. Tym razem jednak postanowiono poprawić opracowany system, gdyż znaleziono w nim niedoróbki. Dodatkowo nastąpił rozwój technologii wytwarzania oprogramowania - odkryto iteratory. W konkursie trzeba będzie sprawdzać pierwszość liczb. Wybrano najprostszą metodę deterministyczną, jednak ze względów optymalizacyjnych postanowiono nie sprawdzać liczb parzystych większych od dwóch. Po głębszych analizach zdecydowano na użycie 2 iteratorów: jednego który przechodzi przez liczby całkowite zgodnie z parametrami, oraz drugi który iteruje po 2 wskazanych iteratorach.

Napisz program który:

1. Otwiera plik tekstowy z liczbami naturalnymi (1 linia, 1 liczba).
2. Wczytuje po kolei wszystkie liczby z pliku i określa ich pierwszość.
3. Implementuje iterator liczb całkowitych z 3 parametrami w konstruktorze: pierwsza liczba, górne ograniczenie[4], oraz skok - o ile należy przeskoczyć.
4. Implementuje iterator, który w konstruktorze przyjmuje 2 inne iteratory i iteruje najpierw przez cały pierwszy, a potem przez cały drugi.
5. Wykorzystuje oba iteratory do przejścia przez liczby 2, 3, 5, 7, 9, ...
6. Oba iteratory powinny implementować interfejs `java.util.Iterator<Integer>`
7. Wypisuje wyniki na ekran.

Zadanie 2. Szybko się okazało, że iteratory to bardzo przydatne narzędzie. Kawuś postanowił zorganizować mały pokaz na którym zaprezentuje odkrycie iteratorów. Zlecił wybranie zagadnienia do prezentacji swoim ludziom. Wybrano iterację po liczbach rzeczywistych oraz przez opakowanie tego przez funkcje.

Napisz program który:

1. Implementuje iterator po liczbach rzeczywistych z 3 parametrami w konstruktorze: pierwsza liczba, górne ograniczenie[5], oraz skok którego wartość jest różnicą między dwoma kolejnymi wartościami.
2. Implementuje iterator po liczbach rzeczywistych który w konstruktorze przyjmuje inny iterator i każdą kolejną wartość oblicza jako `y = sin(x)`, gdzie x to kolejna wartość iteratora podanego z konstruktora.
3. Oba iteratory powinny implementować interfejs `java.util.Iterator<Double>`
4. Wypisuje wyniki na ekran.

[4] liczba która jest większa lub równa największej zwróconej
[5] liczba która jest większa lub równa największej zwróconej

Laboratorium algorytmów i struktur danych

Laboratorium 4

Listy wiązane, stosy

Zadanie 1. Nastały ciężkie czasy. Podczas eksperymentów nad nowatorskim rozwiązaniem paru specjalistów pokłóciło się, które rozwiązanie dla stosu jest lepsze: te na liście wiązanej, czy te na tablicy. Ponieważ konflikt się zaogniał, władca Kawuś VII postanowił rozwiązać spór przez ogłoszenie konkursu, na najlepszą implementację.

Napisz program który:

1. Zawiera 2 implementację stosu: na liście wiązanej (własna implementacja) i na tablicy (można użyć `ArrayList`).
2. Obie implementacje stosu muszą mieć interfejs zgodny z `java.util.Stack<E>`[6], ale nie musi po nim dziedziczyć.
3. Zawiera testy na stosie liczb całkowitych (`Stack<Integer>`).
4. Ww. testy mają sprawdzać wydajność obu implementacji.

Zadanie 2. Wyniki konkursu mają zostać niedługo ogłoszone. Spór przerodził się w oczekiwanie. Jednak, żeby rozluźnić atmosferę wśród ekspertów, Kawuś VII zaproponował, żeby wykonać kalkulator który wykona wyrażenie arytmetyczne. Postanowiono wykorzystać do uproszczenia implementacji, odwrotną notację polską jako pośrednik.

Napisz program który:

1. Pobiera z pliku działania arytmetyczne (jedna linia, jedno działanie) które składają się z liczby rzeczywistej i możliwych działań: dodawania, odejmowania, mnożenia, dzielenia w notacji wrostkowej[7]
2. Przekształca pobrane wyrażenie do ONP[8]
3. Oblicza wynik przekształconego wyrażenia przy użyciu stosu z pierwszego zadania
4. Wyświetla wykonane działanie z wynikiem.

Zadanie 3. (na 5,5) Władca jest zadowolony z wyników swoich specjalistów. Dlatego postanowił przedstawić program innemu zespołowi, który ku jemu zdziwieniu zgłosił problem z brakiem obsługi nawiasów i funkcji matematycznych. Władca postanowił zlecić poprawienie braków.

Napisz program który:

1. Rozszerza implementację z zadania 2go o nawiasy i funkcje jedno-argumentowe: exp, sqrt, sin, cos, tan.
2. Dodanie funkcji ma zakładać możliwość dodania kolejnych w łatwy sposób, np. przez zdefiniowanie interfejsu który implementują klasy z danymi funkcjami[9].

[6] http://docs.oracle.com/javase/7/docs/api/java/util/Stack.html
[7] infiksowej, czyli tej „zwykłej" typu: 2 + 3 * 5
[8] Odwrotna Notacja Polska, notacja przyrostkowa (postfiksowa), np. 2 3 5 * +
[9] Polecam wykorzystać również mapy (np. `TreeMap`)

Laboratorium algorytmów i struktur danych

Laboratorium 5

Sortowanie proste

Zadanie 1. Podczas przygotowywania statystyk dla Kawusia VII zespół specjalistów napotkał problem: jak poukładać dane w określonej kolejności? Dlatego postanowili zrobić rozeznanie i okazało się, że metod jest więcej niż jedna. Władca dowiedział się w między czasie o tym i wyszedł z propozycją, żeby zrobić konkurs na implementację najszybszej metody.

Napisz program który:

1. Implementuje 3 metody sortowania tablicy `int`ów: bąbelkowe (ang. bubble sort), przez prosty wybór (ang. selection sort) i przez proste wstawianie (ang. insertion sort).
2. Porównaj wydajność dla różnej wielkości tablic (np. 1 000 000, 100 000 i 10 000, ale te liczy nie są obligatoryjne).
3. W każdej turze[10] testów różne algorytmy muszą sortować te same dane, ale dla różnych tur muszą być różne zestawy (losowe).
4. Przeprowadzić kilka prób/tur.

Zadanie 2. Władca był zadowolony z wyników. Dlatego postanowił sprawdzić zaimplementowane algorytmy tak, żeby można było je wykonywać na różnych danych.

Napisz program który:

1. Implementuje algorytmy z zadanie #1 w sposób generyczny - z komparatorem.
2. Testuje czasy wykonania sortowania tablicy obiektów. Obiekty mają składać się z 2 `String`ów o różnych priorytetach (jak np. imię i nazwisko).
3. Testy mają być wykonane analogicznie do tych z zadania #1.

[10] Przez turę należy rozumieć jednokrotne odpalenie 3 algorytmów

Laboratorium algorytmów i struktur danych

Laboratorium 6

Sortowanie szybkie

Zadanie 1. Cześć i chwała Władcy! Odkryto rekurencję! W krainie Kawusia VII trwa świętowanie nowego odkrycia. Władca zafascynowany nowym podejściem do sposobu rozwiązywania problemów postanowił zlecić sprawdzenie, w których przypadkach rekurencyjne rozwiązanie zadziała lepiej. Po różnych próbach okazało się, że nie ma zbyt wielu takich problemów. Ostało się sortowanie. Przybity władca wyznaczył nagrodę, za przyspieszenie sortowania przez wykorzystanie rekurencji. Po tytuł zwycięzcy ubiegały się dwa zespoły: Tablicowców i Liściarzy. Każdy zespół z innym podejściem. Każdy z takim samym zapałem do pracy. Tablicowcy sortowali tablice algorytmem `QuickSort` a Liściarze listy wiązane, jednokierunkowe, przy użyciu `MergeSort`.

Napisz program który:

1. Implementuje 2 sortowania: `QuickSort` dla tablicy (`ArrayList`) i `MergeSort` dla listy wiązanej (`LinkedList`).
2. Porównać czasy sortowania obu struktur między sobą o tej samej ilości elementów.
3. Obie wersje sortowania powinny być rekurencyjne.
4. Przeprowadzić wiele testów, dla różnych zestawów danych.

Laboratorium algorytmów i struktur danych

Laboratorium 7

Wyszukiwanie liniowe i binarne

Zadanie 1. W krainie Kawusia VII rozwój technologii trwa w najlepsze. Postęp jest tak szybki, że nie zawsze wiadomo, do czego wykorzystać odkryte możliwości. Czasem również niektóre odkrycia bardzo ciężko uzasadnić. Pojawili się przeciwnicy rozwoju informatyki twierdząc, że programiści korzystają z magii. A wszystko zaczęło się od statystyk na losowo wypełnionych tablicach.

Napisz program który:

1. Tworzy tablicę `int`ów o rozmiarze co najmniej 1 000 000 elementów i wypełnia ją losowymi liczbami. Z zakresu od 0 do 1000.
2. Zawiera funkcję, która dla dowolnej tablicy `int`ów znajduje w niej ile jest liczb mniejszych niż zadana w parametrze (np. są dwa parametry tablica i liczba 128. Wynikiem będzie liczba liczb mniejszych niż 128 w ten tablicy).
3. Mierzy czas wykonania operacji zliczania z #2 punktu dla tablicy z #1 punktu.
4. Sortuje gotową metodą tablicę z pkt #1.
5. Mierzy (ponownie) czas funkcji z punktu #2 dla posortowanej w punkcie #4 tablicy.
6. Korzystając z faktu, że tablica jest posortowana wykonuje wyszukiwanie liniowe zadanej liczby i zwraca go jako wynik jednoznaczny z poprzednimi operacjami i mierzy czas tej operacji.
7. Ponownie korzystając z faktu posortowania tablicy znajduje wyszukiwaniem binarnym zadaną liczbę, po czym najmniejszy indeks w pod którym jest ta lub większa liczba[11] - również mierzy czas operacji.
8. Zestawia czasy i wyniki w czytelny sposób - 4 pary czas-wynik.

[11] Zabezpieczenie przed możliwością wystąpienia tej liczby wiele razy

Laboratorium algorytmów i struktur danych

Laboratorium 8

Drzewa binarne i słowniki

Zadanie 1. Kawuś VII miewa od jakiegoś czasu koszmary. Po długich rozmowach z nadwornym lekarzem wyszło na jaw, że władcę w snach goni litera „X". Zaniepokojeni nadworni postanowili dowiedzieć się więcej i pomóc władcy. Kawuś VII po rozmowach zgodził się na ofertę pomocy i wyznaczył osoby odpowiedzialne za rozwiązanie problemu. Po dłuższym dochodzeniu postanowiono sprawdzić, ile jest wyrazów z literą „X" w dostępnych treściach z myślą, o wyeliminowaniu jej z alfabetu.

Napisz program który:

1. Wczytuje plik tekstowy[12] do pamięci RAM[13] - najlepiej w języku angielskim.
2. Buduje listę niepowtarzających się słów składających się wyłącznie z liter, myślników (minusów) i cyfr.
3. Do każdego słowa musi znaleźć ilość jego wystąpień.
4. Nie wolno korzystać ze struktur drzewiastych i słownikowych - implementacja ma być wykonana na tablicach lub listach.
5. Na koniec podlicza ile wyrazów zawiera literę „X" (bez względu na wielkość) - podać należy liczbę wyrazów niepowtarzających się, jak i sumę powtarzających się i zestawić ze wszystkimi wyrazami (razem 4 wyniki).
6. Policzyć czas operacji zaczynając od momentu po wczytaniu pliku do pamięci do momentu uzyskania wyników.

Zadanie 2. Znajdowanie działa. Jednak pojawiły się problemy wydajnościowe. Po głębszym zbadaniu problemu postanowiono zmienić struktury danych z płaskich, na drzewiaste.

Skopiuj program z zadania #1 i w nim:

1. Zmień implementację tak, żeby wykorzystywała drzewo binarne z kluczem w postaci wyrazu w łańcuchu znaków, oraz wartością jako liczą wystąpień wyrazu.
2. Porównaj czasy z poprzednim zadaniem, dla tych samych plików.

Zadanie 3. Problemy wydajnościowe występują dalej. Przyjrzano się problemowi dokładniej - wniosek jest jeden - musimy zrównoważyć drzewo. Ze względu na dużą presję czasu wybrano metodę prostą i skuteczną: drzewo AVL.

Skopiuj program z zadania #2 i w nim:

1. Zmień implementację tak, żeby wykorzystywała drzewo AVL zamiast zwykłego drzewa binarnego.
2. Porównaj czasy z poprzednim zadaniem, dla tych samych plików.

[12] Gotowe pliki w języku angielskim można znaleźć np. tutaj: `http://www.gutenberg.org/`

[13] UWAGA! Otwarcie pliku nie jest jednoznaczne z jego wczytaniem - plik wczytany powinien znajdować się w całości w zmiennej typu `String`, lub podobnej. `BufferedReader` również nie wczytuje całego pliku!

Laboratorium algorytmów i struktur danych

Laboratorium 9

Drzewa czerwono-czarne

Zadanie 1. Przed pokazem na adeptów alchemików starszy specjalista alchemii eksperymentalnej udzielał instrukcji młodym adeptom. Kilkukrotnie powtarzał, że nie wolno pod żadnym pozorem mieszać czerwonej substancji z czarną mazią. Jednak podczas pokazu doszło do wypadku, w którym zmieszane czarne z czerwonym... i z mazi zaczął się wyłaniać dziwny coś, który przypominał czarne drzewo z czerwonymi liśćmi. Widział to jeden z nadwornych informatyków, który doznał wtedy olśnienia. Wymyślił drzewa czerwono-czarne i postanowił sprawdzić ich cechy.

Napisz program który:

1. Implementuje drzewa czerwono czarne analogicznie do drzew AVL z poprzedniej listy.
2. Realizuje porównanie wybranych cech drzew AVL i czerwono-czarnych. Te cechy to: wysokość drzewa, czas wstawiania dużej liczny elementów, czas wyszukiwania dużej liczby elementów.
3. Zawiera testy pokazujące ww. różnice w wybranych warunkach.
4. Implementacje opierają się o generyczne interfejsy.

Laboratorium algorytmów i struktur danych

Laboratorium 10

B-Drzewa

Zadanie 1. Prowadzenie statystyk słów okazało się bardziej przydatne, niż na początku tego się spodziewano. Pojawił się jednak problem: jak zapisać wyniki indeksowania słów na później, żeby tego ponownie nie liczyć? Władca doszedł do siebie po serii koszmarów i na spokojnie stwierdził, że trzeba mieć możliwość szybkiego znajdowania miejsc wystąpienia danego słowa w tekście. Dlatego zlecił wykonanie stosownych narzędzi swoim informatykom.

Napisz 1 program który:

1. Otwiera duży[14] plik tekstowy.
2. Buduje listę wystąpień słów: dla każdego słowa podaje w których miejscach w pliku wystąpił - np. nr linii i nr znaku w tej linii w której się zaczyna. Słowa na liście nie mają się powtarzać - powtarzać mają się wystąpienia.
3. Listę zapisuje do dodatkowego pliku[15] w taki sposób, żeby znając indeks wpisu można było w złożoności O(1) odczytać listę miejsc wystąpień.
4. Buduje indeks słów i zapisuje go do drugiego dodatkowego pliku w postaci B-Drzewa. Indeks ma służyć dostępowi w czasie O(log n) do listy wystąpień.

Napisz 2 program który:

1. Odczytuje dwa dodatkowe pliki i dla zadanego słowa podaje jego wystąpienia w tekście.
2. Pozwala zweryfikować wyniki.

Uściślenia:

- Jak plik wejściowy nazywa się plik1.txt, to dwa pozostałe mogą się nazywać: plik1.dat i plik1.idx.
- B-Drzewo powinno mieć pełną implementację, ale minimalna wystarczy - będzie to skutkować oceną.
- Możliwość odświeżenia indeksu bez usuwania plików będzie szczególnie nagrodzona.
- Pokazanie czasu wykonania programu 1 i 2 jest obligatoryjne.
- Nie trzeba wczytywać całego pliku do RAMu - dla dużych plików (które są wskazane) nie ma to sensu.

[14] co najmniej 10tys słów
[15] Najlepiej binarnego

Laboratorium algorytmów i struktur danych

Laboratorium 11

Tablice mieszające

Zadanie 1. W krainie Kawusia VII panuje niebywały porządek. Sortowanie i struktury uporządkowane królują. Aż do dzisiaj. Władca wpadł w szał. Ma dość pedantycznego porządku. Nie wyobraża sobie dalej tak idealnej harmonii. Ma panować chaos i nieporządek, „Chociaż na chwilę" jak to władca określił. Przerażeni informatycy postanowili uwolnić krainę od furii władcy i w ramach niespodzianki przedstawić nowe słowniki, które nie będą uporządkowane. Całość była wykonywana w ścisłej tajemnicy. Jednak podczas prac wynikł konflikt. Czołowi specjaliści nie potrafili się zdecydować, które rozwiązanie wybrać. Postanowiono zaimplementować dwa i porównać.

Napisz program który:

1. Implementuje 2 wersji tablicy mieszającej:
   - Z listą wiązaną dla powtarzających się skrótów (ang. hashes) - lista łańcuchowa,
   - Z listą z adresowaniem otwartym - z możliwością prostego parametryzowania skoków;
2. Ma możliwość zmiany rozmiar tablicy (wiąże się to z wygenerowaniem nowej struktury wewnętrznej).
3. Wykonuje operację zliczania słów tak jak na poprzednich listach.
4. Porównuje czasy działania z drzewami.
5. Podpowiedź: warto skorzystać z gotowej metody haszującej w klasie `String`

Zadanie 2. Testy wyglądają bardzo dobrze, więc postanowiono pokazać władcy co wykonano. Władca był zachwycony, ale jak ujawnił sam począł kroki w stronę chaosu i zainteresował się tematem kryptograficznych funkcji skrótu.

Napisz program który:

1. Jest kopią programu z zadania #1, ale wykorzystuje kryptograficzną funkcje skrótu (np. MD5 z api Javy) zamiast użytej wcześniej.
2. Porównuje wyniki wszystkich (4) implementacji tablic mieszających.

Laboratorium algorytmów i struktur danych

Laboratorium 12

Pozycyjne typy danych - optymalizacje

Teoria. We współczesnych komputerach proste typy danych, na których wykonuje się obliczenia można podzielić np. na stałoprzecinkowe i zmiennoprzecinkowe. W Javie zmiennoprzecinkowe są dwa: `float` (32 bity) i `double` (64 bity). Stałoprzecinkowe to: `byte` (1 bajt), `short` (2 bajty), `int` (4 bajty) i `long` (8 bajtów). Typy zmiennoprzecinkowe przechowują reprezentację zaokrąglonych liczb rzeczywistych. Każda taka liczba (zgodna z IEEE754) składa się z 3 części: bitu znaku, części w której zapisany jest wykładnik i części ułamkowej (mantysy). Część ułamkowa zawsze zawiera wartość między 0 a 1. Część wykładnika zawiera informację, gdzie mieści się kropka dziesiętna. Zapis liczy zmiennoprzecinkowej przypomina zapis naukowy tej liczby (6.02E-23 $= 6.02 * 10^{-23}$), z tą różnicą, że podstawą jest 2 a nie 10: 0.101101E1010.

Liczby stałoprzecinkowe przechowują najczęściej wartości całkowite. Jednak przy użyciu paru prostych sztuczek można przechowywać w nich wartości stałoprzecinkowe. Na początku należy założyć, ile bitów jest potrzebnych do przechowywania wartości po przecinku, a ile przed. Np. jak mamy `int`a (który ma 32 bity, z czego 1 na znak), oraz wiemy że największa przechowywana wartość będzie mniejsza od 8 (2^3), to wystarczy nam 3 bity na część całkowita i 28 na ułamkową. Żeby `double` przepisać do takiego `int`a wystarczy zrobić np. tak:

```
double d = Math.rand() * 7.999;
int i = (int)(d * Math.pow(2., 28));
```

Analogicznie zamieniamy liczy w drugą stronę.
Dodawanie i odejmowanie wykonujemy normalnie. Jednak przy mnożeniu i dzieleniu należy wykonać korekty. Mnożąc dwie 32 bitowe liczy wynik jest maksymalnie 64 bitowy, a dodatkowo przy mnożeniu liczb ze znakiem przesuwa się przecinek. Mnożenie z korekcją wygląda np. tak:

```
int a, b, c; // c = a * b;
// ...
c = (int)(((long)a * b) >> 28);
```

Zadanie 1. Mnożenie macierzy kwadratowych na czas - macierze gęste po np. 1 tys. wierzy/kolumn.

Napisz program który:

1. Pozwala wczytać macierz liczb rzeczywistych z pliku tekstowego, albo generują losową - program ma wspierać obie metody.
2. Wybiera 2 macierze i mnoży je licząc czas obliczeń, oraz zapisując wynik do pliku.
3. Ma 3 tryby pracy: `double`, `float`, `int`.
4. Dla ułatwienia niech wartości w macierzach będą w przedziale $< 0, 1)$ - wystarczą 3 bity na część całkowitą.
5. Wyświetla czas obliczeń.

Zadanie 2. Obliczanie wyznacznika macierzy na czas. Napisz program który z takimi samymi założeniami jak zad #1 oblicza wyznacznik macierzy (wczytanej, lub losowej).

Laboratorium algorytmów i struktur danych

Laboratorium 13

Grafy - podstawy

Zadanie 1. Informatyka promowana przez Kawusia VII panoszy się coraz bardziej. Jakiś czas temu wdarła się do zatwardziałego grona kartografów. Nie dużo musiało czasu minąć, żeby informatyka nawet ich przekonała do siebie. Bardzo spodobała się kartografom idea grafów - mogli by nimi reprezentować mapy. Na początek jednak nie do końca widzieli, jak się poruszać po takiej mapie.

Napisz program który:

1. Implementuje 2 algorytmy grafowe: przechodzenie w głąb i w szerz.
2. Zawiera 2 implementację grafów: sieciową (każdy węzeł grafu jest obiektem z listą sąsiadów) i macierzową.
3. Porównuje każdy z 4 wariantów pod względem czasu.
4. Grafy wczytuje z plików.
5. Dodatkowo punktowane jest pomysłowe i sensowne losowanie grafów.