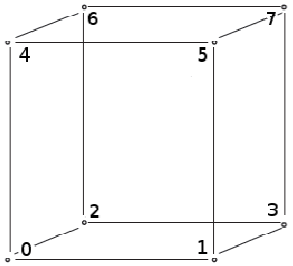
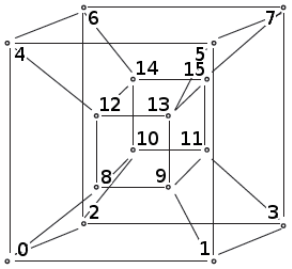
Praca domowa 09 – hypercube

Termin zwrotu : 12 stycznia godz. 23.00
Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

W projektowaniu algorytmów równoległych istotną grupę stanowią architektury określane mianem *'hypercube'*. N-wymiarowy hypercube (jako architektura systemu wieloprocesorowego) definiowany jest jako zbiór wierzchołków n-wymiarowej kostki jednostkowej. Numer wierzchołka (procesora) wyznaczany jest przez wartość ciągu binarnego reprezentującego współrzędne określonego wierzchołka, a więc np. w przestrzeni 4-wymiarowej procesor o współrzędnych (1,1,0,0) oznaczać będziemy jako procesor o numerze 12. Dwa procesory połączone są pomiędzy sobą łączem komunikacyjnym wtedy i tylko wtedy, gdy wektory ich współrzędnych różnią się wyłącznie na jednej pozycji. Np. procesor 12 (1,1,0,0) połączony będzie z procesorem 4 (0,1,0,0) , 8 (1,0,0,0) , 14 (1,1,1,0) oraz 13 (1,1,0,1). Na rysunkach przedstawiono przykłady architektur 3-cube oraz 4-cube. Komunikowanie się z otoczeniem zewnętrznym możliwe jest wyłącznie poprzez węzeł 0 (czyli np. wprowadzanie i początkowa dystrybucja danych, oraz wyprowadzanie wyników).





Należy zdefiniować co najmniej klasy : *Hypercube* umożliwiającą utworzenie (skonfigurowanie) architektury o N węzłach, gdzie N określone jest parametrem `<n-size>` programu oraz *Node* opisująca działanie pojedynczego węzła (każdy węzeł wykonywać winien swoje działania jako oddzielny task). Klasa *Hypercube* winna posiadać dwie abstrakcyjne metody publiczne (*put()*, *get()*). Implementacja konkretnej metody *put* winna umożliwiać specyficzną dla rozwiązywanego zadania procedurę wprowadzenia i redystrybucji danych, metody *get* – pobranie wyniku obliczeń. Publiczna metoda *run()* klasy *Hypercube* winna uruchomić proces obliczeniowy w 'całej architekturze', nadzorując proces (moment) jego zakończenia. Abstrakcyjna metoda *run()* klasy *Node* umożliwia implementację realizowanego przez każdy z węzłów algorytmu równoległego (a więc każdy z węzłów realizuje ten sam program na innych danych !).

W pliku `<input_file>` zapisanych jest k*N liczb rzeczywistych, gdzie k > 1. Należy zaimplementować równoległy algorytm sumowania zbioru liczb działający w ilości kroków O(k).

Program może być zapisany w kilku plikach przy czym struktura plików musi być płaska (wszystkie pliki w jednym katalogu). Program nie może korzystać z jakichkolwiek bibliotek zewnętrznych. Plik `Hypercube.java` zawierać musi implementację klasy *Hypercube*, `Node.java` implementację klasy *Node*, `Algorithm.java` implementacje algorytmu obliczeń (a więc np. implementację metod *put* i *get* w interfejsie *IHypercube* oraz metody *run* w interfejsie *INode*). `Main.java` zawierać musi zapis dynamicznego tworzenia obiektu reprezentującego architekturę równoległą (N-cube).

Proces kompilacji musi być możliwy z użyciem komendy

```
javac –Xlint *.java
```

Uruchomienie programu winno być możliwe z użyciem komendy

```
java Main <input_file> <n-size>
```

Wynik końcowy (w strumieniu wyjściowym nie powinny pojawiać się jakiekolwiek inne elementy – np. wydruki kontrolne) działania programu musi zawierać pojedynczą liczbę określającą wartość poszukiwanej sumy (dokładność do 5 miejsc dziesiętnych), a więc np.

```
Suma : 23.93640
```

Wymagania :

- Klasy implementujące problem winny zostać zdefiniowane w plikach `Hypercube.java, Node.java, Algorithm.java`
- Klasa implementująca mechanizm program główny (metoda main) winny być zdefiniowane w pliku `Main.java`
- W pliku README.pdf winien być zawarty szczegółowy opis organizacji struktur danych oraz szczegółowy opis zastosowanego algorytmu obliczeniowego.
- Proces obliczenia rozwiązania winien się kończyć w czasie nie przekraczającym 1 min (orientacyjnie dla typowego notebooka). Po przekroczeniu limitu czasu zadanie będzie przerywane, i traktowane podobnie jak w sytuacji błędów wykonania (czyli nie podlega dalszej ocenie).

Sposób oceny **:**

- 1 pkt – **Kompilacja** : każdy z plików winien być kompilowany bez jakichkolwiek błędów lub ostrzeżeń (w sposób omówiony wyżej)
- 1 pkt – **Wykonanie** : program powinien wykonywać się bez jakichkolwiek błędów i ostrzeżeń (dla pliku danych wejściowych zgodnych z wyżej zamieszczoną specyfikacją) z wykorzystaniem omówionych wyżej parametrów linii komend
- 2 pkt – **README** : plik README.pdf dokumentuje w sposób kompletny i właściwy struktury danych, oraz opis przyjętej koncepcji algorytmu równoległego
- 1 pkt – **Komentarze wewnętrzne** : czy program jest skomentowany w sposób zapewniający zrozumienie jego działania, oraz wyjaśniający warunki, które muszą zachodzić przed i po wykonaniu każdej z funkcji.
- 1 pkt – **Styl kodowania** : czy funkcji i zmienne posiadają samo-wyjaśniające nazwy ? Czy podział na funkcje ułatwia czytelność i zrozumiałość kodu ? Czy funkcje eliminują (redukują) powtarzające się bloki kodu ? Czy wcięcia, odstępy, wykorzystanie nawiasów itp. (formatowanie kodu) są spójne i sensowne ?
- 4 pkt – **Poprawność algorytmu** : czy algorytm został zaimplementowany poprawnie a wynik odpowiada prawidłowej (określonej zbiorem danych testowej) wartości.