1. Wprowadzenie do biblioteki STL.
2. Reprezentacja grafów.
3. Zadanie S4 jako przykład problemu, w którym przydatna jest biblioteka STL.
4. Polecane książki:

- Biblioteka Standardowa C++. Podręcznik programisty
-

STL w praktyce. 50 sposobów efektywnego wykorzystania

Zadanie domowe

1. Rozwiązać zadanie S4 .

Użyteczne klasy i funkcje

- <algorithm>: sort, reverse, binary_search, lower_bound, upper_bound, next_permutation, unique, find, search, min, max, swap
- <cctype>: isdigit, islower, isupper, isalpha, isalnum, isspace, tolower, toupper
- <map>: słownik - użycie:
  map<string, int> slownik;
  slownik[string("foo")] = 7;
- <queue> - kolejka zwykła oraz priorytetowa - użycie:
  queue<int> q;
  priority_queue<int> kolejka_prio;
  priority_queue<pair<int, int>, vector<pair<int, int> >, greater<pair<int, int> > > kolejka_do_kopca;
  q.push(17);
  while (!q.empty()) { int a = q.front(); q.pop(); }
  Dla kolejki priorytetowej zamiast front() jest top().
- <set>: zbiór - użycie:
  set<int> zbior;
  zbior.insert(17);
  if (zbior.find(15) != zbior.end()) printf("Element znajduje sie w zbiorze");
- <sstream>: parsowanie łańcuchów tak jak cin, cout - uwaga: dość wolne - użycie:
  istringstream wejscie("10 20 30");

int t[3];
for (int i = 0; i < 3; i++) wejscie >> t[i];
Istnieje też ostringstream

- <string>: proste w użyciu łańcuchy znaków
konwersja na string: string tekst = string("blabla");
konwersja na char*: printf("%sn", tekst.c_str());
- <utility>: pary liczb, porównywanie - pierwasy element, jak równe, to drugi - użycie:
pair<int, double> para = make_pair(10, 12.5);
para.second += para.first;
- <vector>: tablica o zmiennej długości - użycie:
vector<int> tab;
tab.clear();
tab.push_back(17);
tab.push_back(15);
tab[1] += tab[0];
tab.pop_back();

Uwaga: dodawanie n elementów powoduje wykonanie około 1.5 * n operacji wstawienia, aby tego uniknąć należy podać rozmiar tablicy w konstruktorze: vector<int> tab(100000) lub użyć funkcji tab.resize(100000). Tak zmodyfikowany wektor działa tak samo jak tablica o 100000 elementów. Dodanie elementu za pomocą push_back() wstawi go na pozycję 100001.

Przydatne kody

Przeglądanie wektora może być zrealizowane na dwa sposoby. W pierwszym krótszy jest kod w instrukcji for(), ale dłuższe odwołania do elementu, w drugim odwrotnie. Uwaga na operator "!=":

```
vector<int> tab;
for (unsigned int i = 0; i < tab.size(); i++) dowork(tab[i]);
for (vector<int>::iterator i = tab.begin(); i != tab.end(); i++) dowork(*i);
```

Usuwanie powtarzających się elementów z wektora:

```
vector<int> tab;
sort(tab.begin(), tab.end());
```

```
tab.erase(unique(tab.begin(), tab.end()), tab.end());
```

Generowanie wszystkich permutacji:

```
for (int i = 0; i < n; i++) perm[i] = i;
do {
dowork(perm);
} while (next_permutation(perm, perm + n));
```

Uwagi

Dwa nawiasy ">"muszą być odzdzielone spacją: vector<pair<int, int>[spacja]>

Strumienie (cin, cout) są bardzo wolne w porównaniu ze stdio.h. Często na zawodach algorytmicznych samo wczytywanie danych za ich pomocą może się nie zmieścić w limicie czasu. Można je trochę przyspieszyć umieszczając na początku programu linijki:

```
ios_base::sync_with_stdio(false);
cin.tie(0);
```