Zadanie F - Firma kurierska 5 (Decomposition + 2 opt i node-insertion)

Firma Kurierska rozwożąca paczki w Zjednoczonych Stanach Bajtocji, w której pracujesz bardzo się rozbudowała. Po ostatnim kryzysie nie ma już śladów. Twój ostatni program bardzo poprawił jej zyski. Równocześnie stała się najbardziej znaną i cenioną firma kurierską w całej Bajtocji. Obsługuje coraz większą liczbę klientów. Bywają dni, kiedy liczba przesyłek do rozwiezienia jest tak duża, że algorytmy wyznaczające trasy dla kurierów działają baaardzo długo, przez co kurierzy wyjeżdżają w trasę z opóźnieniem. Postanowiłeś napisać nowy program, który będzie dedykowany dla tych sytuacji.

Nowy program wykorzysta technikę zwaną *Rectangle Decomposition*, która dzieli cały graf na k^2 prostokątów, każdy niepusty prostokąt traktuje jak 1 wierzchołek i wyznacza trasę T dla nowego grafu. Następnie program liczy trasy dla każdego z prostokątów z osobna i policzone trasy łączy w jedną zgodnie z kolejnością występowania danego prostokąta w trasie T. Trasy wyznaczone w programie są poprawiane metodą *2-opt*.

Podobnie jak w poprzednich programach, wykorzystasz fakt, że długości autostrad w Zjednoczonych Stanach Bajtocji łączących dwa miasta w przybliżeniu odzwierciedlają rzeczywistą odległość tychże dwóch miast. Możesz więc ograniczyć się do euklidesowego problemu komiwojażera i korzystać z grafu kandydatów obliczonego na podstawie grafu Delaunaya.

Wejście

Pierwsza linia wejścia zawiera liczbę całkowitą z ($1 \leqslant z \leqslant 2 * 10^9$) – liczbę zestawów danych, których opisy występują kolejno po sobie. Opis jednego zestawu jest następujący:

W pierwszej linii zestawu znajduje się liczba naturalna n oznaczająca liczbę miast opisanych jako punkty na płaszczyźnie ($2 \leqslant n \leqslant 200000$). W kolejnych n liniach znajdują się współrzędne punktów, tzn. dwie liczby rzeczywiste a i b.

Wyjście

Dla każdego zestawu danych w pierwszej linii należy wypisać permutację miast (miasta numerowane są od 0), opisującą zalecaną trasę dla kuriera rozwożącego paczki. W drugiej linii należy wypisać długość znalezionej trasy.

Dostępna pamięć: 256MB

Uwagi o rozwiązaniu wzorcowym

- Grafy kandydatów wyznaczane w programie są sumą triangulacji Delaunay'a oraz *10-nearest neighbours.*
- Algorytm pracuje na k^2 prostokątach o równych wysokościach i szerokościach, gdzie $k = \sqrt[4]{n}$ (za wyjątkiem prostokątów brzegowych). Niepuste prostokąty stanowią wierzchołki meta-grafu.
- Odległość między dwoma niepustymi prostokątami (wierzchołkami meta-grafu) liczona jest jako odległość między dwoma najbliższymi punktami takimi, że jeden należy do otoczki punktów prostokąta A, a drugi do otoczki punktów prostokąta B. Wyznaczone odległości

pamiętane są w macierzy sąsiedztwa. Dla każdej pary prostokątów ich odległość liczona jest w algorytmie co najwyżej jeden raz (gdy jest potrzebna).

- Trasa T wyznaczona dla meta-grafu obliczona jest za pomocą pełnej techniki *Farthest Insertion*, a następnie poprawiana za pomocą ruchów *2-opt* lub *node-insertion* (dopóki taki ruch można wykonać).

- Wewnątrz każdego niepustego prostokąta liczona jest ścieżka komiwojażera miedzy dwoma punktami brzegowymi, za pomocą których dany prostokąt łączy się z innymi w trasie T. Ścieżki te obliczone są pomocą techniki *Farthest Insertion* (w przy wykorzystaniu grafu kandydatów), następnie poprawiane za pomocą ruchów *2-opt* i *node-insertion* (dopóki taki ruch można wykonać).

- Trasa wynikowa jest sumą trasy w meta-grafie i ścieżek wyznaczonych dla prostokątów.

- Na zakończenie uzyskana trasa jest poprawiana za pomocą ruchów *2-opt* i *node-insertion*. Ze względu na ograniczenia czasowe, każdy punkt testowany jest tylko *dwa* razy jako kandydat dla ruchu *2-opt* lub *node-insertion*.

- Poprawianie tras za pomocą ruchów *2-opt* i *node-insertion* wykonywane jest przy wykorzystaniu *drzewa splay* (patrz zadanie D).

- Odległość dwóch miast dla uproszczenia liczona jest za pomocą funkcji:

```
vector<Delaunay::Point> Points;
inline int euclid(int v, int w) {
  double x = Points[v].x() - Points[w].x();
  double y = Points[v].y() - Points[w].y();
  return floor(sqrt(x*x+y*y) + 0.5);
}
```

- Jakość wyznaczonej trasy o długości d mierzona jest względnym odchyleniem od *lower* - długości optymalnej trasy (długości te znane są dla testów z tsplib i tspart) lub (gdy wartości optymalne są nieznane) ograniczeń dolnych na długości optymalnych tras (ograniczenia obliczone metodami *nearest neighbors* i *metodą geometryczną*). Jakość ta liczona jest według wzoru $100 \cdot \frac{d-lower}{lower}\%$. Jakości tras z jednego pliku wejściowego są uśredniane. Dopuszczane są poprawne rozwiązania o średniej jakości co najwyżej 1.5-2% gorszej od jakości rozwiązań wzorcowych. Progi dla kolejnych plików wejściowych prezentuje poniższa tabelka

Pliki wejściowe:	Progi:
dec-0	15
dec-tsplib1	12.5
dec-tsplib2	13
dec-tsplib3	14
dec-tsplib4	12
dec-tsplib5	8.3
dec-tspart1	8
dec-tspart2	8.3
dec-1	22.5
dec-2	22.5

Przykład

Dla danych wejściowych:	Poprawną odpowiedzią jest:
2 6 0 0 1 0 0 1 1 1 0.5 0.5 2 1 16 0 0 0 5 0 10 0 15 1 0 1 5 1 10 1 15 3 0 3 5 3 10 3 15 5 0 5 5 5 10 5 15	0 2 4 3 5 1 6 0 4 8 12 9 13 14 10 15 11 7 3 6 2 5 1 46