Zarys rozwiązań (skopiowany z chatu) :

Zadanie 1

Algorytm można podzielić na 2 części:
1. Wyznaczenie "zakresu przeszukiwania" - sztuczne minimum 2. Przeszukiwanie właściwe 1. Sortujemy listę współrzędnych po współrzędnej 'x' quick sortem/ merge sortem , sprawdzamy pary kolejnych elementów tablicy (k,k+1 dla k od 1 do n-1) wyznaczając najmniejszą odległość między tymi parami. Analogicznie po 'y'. Wybieramy z tych Całość zabiera na O(2*(n*log(n) + n)) czyli O(n*log(n))

2. Mamy po poprzedniej operacji listę posortowaną po 'y', zatem teraz patrzymy na wszystkie następne elementy w odległości co najwyżej oddalone o naszą wcześniej wyznaczoną odlegość minimalną, bądź mniejszą, o ile została po drodze taka wyznaczona, a to powinno zajmować O(k*n) o ile współrzędne rozrzucone są losowo i jest ich dużo.
A gdzie tu divide and conquer ? A no przy sortowaniu w części

Zadanie 2

Master Theorem FTW.

Zadanie 3

Radix sort.
113 jest stałą, reszta do wydedukowania.

Zadanie 4

A co do czwartego to robisz algorytm zachłanny, wzbogacasz wierzchołki grafu o pola z informacją o długości najkrótszego wierzchołka oraz ilości takich wierzchołków, po czym wychodzisz z wierzchołka u, no i iterujesz po incydentialnych wierzchołkach, i rekrencyjnie dla każdego wierzchołka spowrtem wywołujesz dla wszystkich wywołanych, jak się uprzeć to można przechowywać jeszcze w wierzchołkjach skad się wychodziło, ale długość ścieżki eliminuje nam nawroty, co więcej, może się okazać że może w niektórych sytuacjach ratować najlepszą ścieżkę w sytuacji gdy doszliśmy wpierw z gorszego miejsca do pewnego wierzchołka.
No i właśnie wywołania sprawdzenia dla wierzchołków incydentialnych się wykonuje w przypadku jeśli została polepszona najlepsza ścieżka