Graf:

Graf skonstruowany z Nodów odzwierciedlających przystanki

Każdy Node ma przypisaną listę kursów, Edges, wychodzących z tego przystanku

Dijkstra:

Do kolejki priorytetowej przydzielam krotki zawierające czas przyjazdu na przystanek i Node odzwierciedlający ten przystanek; początkowa krotka posiada wartość czasu podaną przez użytkownika oraz startowy przystanek

Za pomocą **heap queue algorithm** (heapq.heappop(pq)) wyznaczam przystanek, do którego najszybciej można się dostać

Jeśli ten przystanek jest poszukiwanym przystankiem, algorytm się kończy

Dla każdego autobusu/tramwaju wyjeżdżającego z wybranego przystanku, o ile czas wyjazdu nie jest wcześniejszy niż czas przyjazdu na ten przystanek, oraz czas dojazdu na kolejny przystanek jest wcześniejszy niż obecny dla tego nowego przystanku, jest dopisywany na listę priorytetową

Na końcu algorytm ponownie wyszukuje kolejny przystanek za pomocą heapq i Dijkstra wraca do punktu 3

A\*:

Algorytm bazujący na Dijkstrze, z tą różnicą, że w kolejce priorytetowej elementem kluczowym dla sortowania jest suma odległości pomiędzy przystankiem końcowym a możliwym przystankiem i czasu dojazdu na ten przystanek

A\* z ograniczeniem przesiadek:

Algorytm A\* puszczony dwukrotnie, przy obu iteracjach, premiowane jest korzystanie z tych samych linii, za drugim razem premiowane jest wczesne znalezienie końcowej linii

Ulepszenie podpunktu b) c):

Zamiast odległości pomiędzy przystankiem końcowym a możliwym przystankiem, pod uwagę brany jest kwadrat, żeby zaoszczędzić na czasie pierwiastkowania