ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ

Мэдээлэл холбоо технологийн сургууль

БИЕ ДААЛТЫН ТАЙЛАН

Алгоритмын шинжилгээ ба зохиомж (F.CSM301)
2024-2025 оны хичээлийн жилийн намар

Хичээл заасан багш:	Д.Батмөнх /F.CS21/
Бие даалтын ажил гүйцэтгэсэн:	Э. Төгсбаяр /B221910021/

Улаанбаатар хот
2025 он

АГУУЛГА

ОРШИЛ

Энэхүү бие даалтын зорилго нь графын хайлтын гурван алгоритм болох BFS, DFS, болон Dijkstra алгоритмуудын үндсэн зарчмыг ойлгож, Улаанбаатар хотын OpenStreetMap (OSM) газрын зураг дээр хоёр цэгийн хоорондох замыг тооцох систем боловсруулах явдал юм Орчин үеийн хотын замын сүлжээ нь өдөр бүр олон мянган хүн, тээврийн хэрэгслийн урсгалыг үйлчилдэг. Улаанбаатар хотын хувьд замын түгжрэл, хамгийн богино замыг олох асуудал нь амьдралын чанарт шууд нөлөөлдөг. Энэхүү судалгааны ажил нь график онолын алгоритмуудыг ашиглан бодит замын өгөгдөл дээр хамгийн оновчтой замыг олох арга зүйг хөгжүүлэхэд чиглэгдсэн.

Алгоритмуудын онол

BFS (Breadth-First Search) алгоритм BFS буюу өргөнөөр хайх алгоритм нь графыг төвшин бүрээр (level by level) хайдаг алгоритм юм. Энэ нь queue (дараалал) өгөгдлийн бүтцийг ашигладаг.
Үндсэн санаа: Эхлэх оройгоос эхлэн төвшин бүрийн бүх оройг үзсэний дараа дараагийн төвшин рүү шилжинэ. Ингэснээр хамгийн цөөн алхамтай замыг олдог.
Алгоритмын алхамууд:
Эхлэх оройг queue-д хийнэ
Эхлэх оройг үзсэн гэж тэмдэглэнэ
Queue хоосон болтол дараах үйлдлүүдийг давтана: • Queue-с орой гаргаж авна • Хэрэв энэ нь зорилтот орой бол зогсоно • Үзээгүй бүх хөрш оройг queue-д хийж, үзсэн гэж тэмдэглэнэ

Давуу тал: • Хамгийн цөөн алхамтай замыг баталгаатай олдог • Түргэн ажилладаг (энгийн давталт) • Хэрэгжүүлэхэд энгийн

Сул тал: • Замын жинг тооцдоггүй • Их санах ой шаарддаг (бүх төвшний оройг хадгалах) • Хамгийн богино зайтай зам олохгүй Хугацааны хүндрэл: O(V + E) Санах ойн хүндрэл: O(V) Энд V нь оройн тоо, E нь ирмэгийн тоо.

DFS (Depth-First Search) алгоритм
DFS буюу гүнд хайх алгоритм нь графыг аль болох гүн рүү явж, цаашид явах боломжгүй болсон тохиолдолд буцаж ирдэг алгоритм юм. Энэ нь stack өгөгдлийн бүтэц эсвэл recursion ашигладаг.
Үндсэн санаа: Нэг чиглэлд аль болох гүн рүү явж, цаашид явах боломжгүй болохоор буцаж ирээд өөр зам турших. Ингэснээр олон хувилбарын зам олох боломжтой.
Алгоритмын алхамууд:
Эхлэх оройг үзнэ
Үзээгүй хөрш орой уруу шилжинэ
Хэрэв хөрш орой байхгүй бол эргэж буцна (backtrack)
Зорилтот орой олох эсвэл бүх оройг үзтэл давтана

Давуу тал: • Бага санах ой шаарддаг • Боломжит олон замыг олох боломжтой • Recursion ашиглан энгийн хэрэгжүүлнэ

Сул тал: • Хамгийн богино зам олохгүй • Мөчлөг (cycle) байвал төгсгөлгүй давтагдаж болно • Илүү удаан байж болно Хугацааны хүндрэл: O(V + E) Санах ойн хүндрэл: O(V)

Dijkstra алгоритм Dijkstra алгоритм нь 1956 онд Холландын компьютер эрдэмтэн Эдсгер Дейкстра зохиосон алгоритм бөгөөд жинтэй графт эхлэх оройгоос бусад бүх орой уруу хамгийн богино замыг олдог.
Үндсэн санаа: Эхлэх оройгоос бусад орой уруу хүрэх хамгийн бага зайг дараалан олох. Priority queue ашиглан хамгийн бага зайтай оройг сонгон, түүний хөршүүдийн зайг шинэчилдэг.
Алгоритмын алхамууд:
Эхлэх оройн зай = 0, бусад бүх оройн зай = ∞ 2. Бүх оройг үзээгүй гэж тэмдэглэнэ
Үзээгүй оройнуудаас хамгийн бага зайтайг сонгоно
Сонгосон оройн бүх хөрш оройн зайг шалгаж, шинэчилнэ: • Хэрэв шинэ зам нь богино бол зайг шинэчилнэ
Сонгосон оройг үзсэн гэж тэмдэглэнэ
Бүх оройг үзтэл эсвэл зорилтот орой олдтол давтана

Давуу тал: • Жингийн хувьд хамгийн богино замыг баталгаатай олдог • Практик маш өргөн хэрэглэгддэг • Оновчтой хэрэгжүүлэлттэй

Сул тал: • BFS-ээс бага зэрэг удаан • Сөрөг жинтэй ирмэгт ажиллахгүй • Илүү их санах ой шаарддаг Хугацааны хүндрэл: O((V + E) log V) Санах ойн хүндрэл: O(V)

Алгоритмуудын хэрэгжүүлэлт

BFS хэрэгжүүлэлт:

```
def bfs(self, start, goal):
    queue = deque([start, [start], 0])
    visited = {start}
    while queue:
        current, path, dist = queue.popleft()
        if current == goal:
            return path, dist
            for neighbor, weight in graph.get_neighbors(current):
                if neighbor not in visited:
                    visited.add(neighbor) queue.append((neighbor, path +
                    [neighbor], dist + weight))
    return [], 0
```

DFS хэрэгжүүлэлт:

```
def dfs(self, start, goal, max_paths = 10):
    all_paths = []
    visited = set()
    def dfs_recursive(current, path, distance):
        if current == goal:
            all_paths.append((path.copy(), distance))
            return
            visited.add(current)
        for neighbor, weight in graph.get_neighbors(current):
            if neighbor not in visited:
                path.append(neighbor)
                dfs_recursive(neighbor, path, distance + weight)
                path.pop()
        visited.remove(current)
    def_recursive(start, [start], 0)
    return sorted(all_paths, key=lambda x: x[1])
```

Dijkstra хэрэгжүүлэлт:

```
def dijkstra(self, start, goal):
    pq = [(0, start, [start])] distances = {start: 0}
    visited = set()
    while pq:
        current_dist, current, path = heapq.heappop(pq)
        if current in visited:
            continue
        visited.add(current)
        if current == goal:
            8 return path, current_dist
        for neighbor, weight in graph.get_neighbors(current):
            if neighbor not in visited:
                new_dist = current_dist + weight
```

```
                    if neighbor not in distances or \ new_dist < distances[neighbor]:
                    distances[neighbor] = new_dist heapq.heappush(pq, (new_dist,
                    neighbor, path + [neighbor]))
        return [], 0
```

Дүгнэлт

BFS: хамгийн цөөн алхамтай замыг хурдан олдог.
DFS: нэг чиглэлд гүн судалдаг тул оновчгүй зам гаргаж болдог.
Dijkstra :замын жинг харгалзан хамгийн богино замыг бодитой гаргадаг. Иймд бодит газрын зураг дээрх зам тооцоололд Dijkstra алгоритм хамгийн тохиромжтой гэж дүгнэв.