

ICE3020 알고리즘설계

<term project>

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2024년 06 월 22 일

학부 정보통신공학

학년 3

성명 이상혁

학번 12201928



1. 개요

Shorth path(최단경로) 문제를 3가지 방법으로 Graph pptx 64 그래프를 통하여 구현을 하였으며 이후에 N x M 형태의 1,000개 이상의 노드 그래프를 5개 이상으로 구성하여 테스트해보았다.  
각 그래프마다 5번씩 체크하여 성능을 확인하였다.

1. 상세 설계 내용

TERM\_PROJECT/

├── astar.h # A\* 알고리즘 선언

├── dijkstra.h # 다익스트라 알고리즘 선언

├── GA.h # 유전 알고리즘 선언

├── astar.cpp # A\* 알고리즘 구현

├── dijkstra.cpp # 다익스트라 알고리즘 구현

├── GA.cpp # 유전 알고리즘 구현

├── task1.cpp # 메인 파일 1

└── task2.cpp # 메인 파일 2

위와 같은 내용으로 프로젝트를 설계하여 진행하였다.

**Dijkstra.h**

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

알고리즘을 불러오기 위한 헤더파일로써의 선언을 진행하였다.

**Dijkstra.cpp**

텍스트, 전자제품, 디스플레이, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 시계이(가) 표시된 사진

자동 생성된 설명

주석을 통하여 세세하게 구현방식을 작성하였다. 우선순위 큐를 이용하여 다음에 탐색할 노드를 결정하게 된다. 현재 노드에서 다음번의 노드로 이동하는데 필요한 비용이 최소인 노드를 우선으로 탐색하기 위함이다. 각 노드까지의 최단거리는 dist 벡터에 저장된다. 이후 최단경로를 검색하기 위해 prev 벡터에는 현재의 노드로 오기전의 노드인덱스가 저장된다. 마지막에는 이 저장한 노드로 start 부터 goal까지의 경로를 출력한다.

**astar.h**

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**astar.cpp**

텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

A\*알고리즘의 핵심은 start에서 goal까지의 최단 경로를 찾을 때 휴리스틱 함수를 활용하여 각 노드에서 목표 노드까지의 추정 거리를 계산 진행하고 이를 기반으로 우선순위를 결정하여 진행하는 점이다. 따라서 휴리스틱을 계산하는 함수의 후보지가 유클리디안, 맨해튼 거리로 두개가 존재하였는데 이때 격자 모양의 지도에서 다른점까지 수평이나 수직으로 이동할때의 거리를 빠르게 계산할 수 있는 맨해튼 기법을 선택하게 되었다. 이후 구현방법은 pq를 이용하여 우선적으로 탐색될 노드를 선택 후 path(경로)를 계산하게 된다.

**GA.h**

소프트웨어, 멀티미디어 소프트웨어, 텍스트, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**GA.cpp**

텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

1. cacluateFitness  
   개체의 염색체를 이용하여 시작 노드에서부터 목표 노드까지의 경로를 확인하고 적합도를 계산한다. 이후 이동할 수 없는 노드가 존재한다면 패널티를 적용한다. 이후 반환
2. createPopilation  
   인구를 생성한다. 무작위 경로인 염색체를 통해 시작 노드에서 목표 노드로 가는 경로 중 하나를 의미하게 된다.
3. 룰렛 selection  
   룰렛 선택 알고리즘을 사용하여 개체들을 선택하게 된다. 확률은 아까 선정한 적합도의 역수에 비례한다.
4. Crossover  
   한 쌍의 부모 개체로부터 교차를 진행한다. 이후 자손이 생성된다. 인간의 유전자처럼 부모의 염색체에서 부분적으로 유전자를 교환하여 진행된다.
5. Mutate  
   염색체 내에서 무작위로 유전자를 다른 유전자로 변경한다. 변이를 적용. 유전적 다양성을 유지하게 된다.
6. geneticAlgorithm  
   유전 알고리즘의 전체과정을 진행하게 된다. 초기 인구를 생성 후 여러 번을 걸쳐 세대에 선택, 교차 ,변이를 진행 해 최적의 해를 가져오게 된다.

Encoding -> Chromosome Representation 정수 인코딩을 진행하였다. 각 유전자는 그래프의 노드를 가리키게 되며 염색체는 노드에서 노드로 이동하는 경로를 나타내는 시퀀스를 의미하게 된다.

Selection -> Roulette Wheel Selection 룰렛 휠 선택을 사용하였다. 강노에 적혀있는 것처럼 적합도가 높은 개체가 선택될 확률이 높아지는 방법이다. 적합도가 높을수록 선택될 확률이 높아지게 된다.

Crossover -> PMX를 사용하였다. 두 부모 염색체에서 교차지점을 무작위로 선택하게 된다. 그 사이에 부분을 교환한다. 이때 염색체를 교환하고 충돌을 피하기 위해 중복된 노드를 제거하도록 하였다

Mutation -> Node swap mutation을 사용하였다. 염색체 내에서 무작위로 하나의 유전자를 선택하고 그 노드의 이웃중 하나를 무작위로 변경한다. 경로의 다양성이 증가되었고 탐색 공간이 증가한다.

Scaling은 적합도를 기반으로 선택하였다. 경로가 더 짧을수록 적합도가 높기에 좋은 경로가 선택되도록 유도가 된다. 따라서 적합도 계산에서 경로의 길이에 따라서 평가하도록 하였다.

**파라미터 설정**

**Population Size: 50**

초기 인구의 크기를 설정.

**Chromosome Length: 10**

각 염색체의 길이를 설정합니다. 염색체는 경로를 나타낸다. 길이가 길수록 노드의 탐색량이 증가한다.

**Generations: 100**

알고리즘이 반복되는 세대 수를 설정

**Mutation Rate: 0.1**

변이가 발생할 확률을 설정

**Start Node: 0**

경로 탐색의 시작 노드를 설정

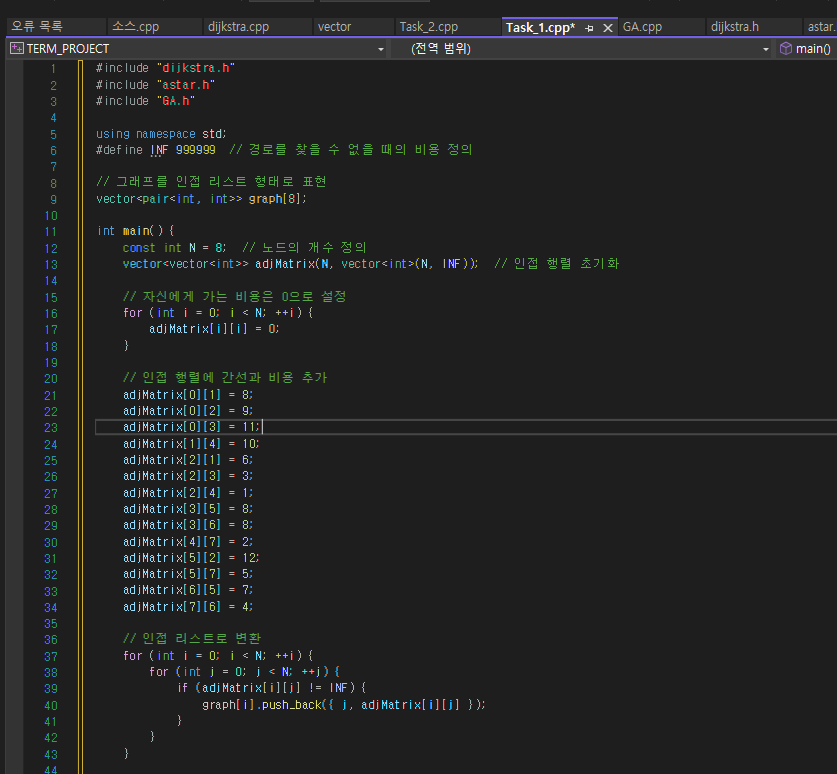
**장점**

1. 효율적 탐색  
   큰 공간이 존재한다면 효율적으로 해를 찾을 수 있다. 초기 인구를 이용하여서 경로를 동시에 탐색할 수 있기 때문이다.
2. 해가 다양  
   교차, 변이를 이용하여 다른 최단경로 알고리즘보다 여러의 해를 생성할 수 있으므로 local minima에 빠질 확률이 줄어든다.
3. 여러형태의 비용함수 사용
4. 병렬 처리  
   각 개체가 동시에 계산이 진행되어 병렬처리가 적합하다.

**단점**

1. Cost  
   파라미터의 설정에 따라 계산 cost가 증가할 수 있다. 따라서 주어진 데이터셋에 맞게 알맞게 조절해야 한다.
2. 해가 local minma에 빠질 수 있다. 항상 동일한 결과를 낳는다는 보장을 할 수가 없다.
3. 다양한 파라미터가 존재해 민감하다.
4. 주어진 데이터셋과 문제에 따라 문제의 해를 구하는데 적합하지 않을 수 있다.

**Task\_1.cpp**



텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

우선 64페이지의 그래프 형태대로 인접행렬을 작성한다음 그래프로 변경하여 테스트를 진행하였다.

유전자알고리즘의 파라미터는 다음과 같이 설정하였음을 확인할 수 있다.

**Task\_2.cpp**

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

랜덤시드를 이용하여 1300개의 노드와 3000개의 간선을 가지는 2차원 백터 배열을 5개 생성하였다. 이후 시간 측정과 함께 각 알고리즘을 0부터 50까지 최단경로를 찾는 방법으로 코드를 작성하였다. 출력형식은 각 그래프에 대해 알리즘 이름과 실행 시간을 ms 단위로 출력하고자 하였으며 각 알고리즘 실행 결과 사이를 구분 짓도록 하였다.

따라서 원래 목적인 다양한 그래프에 대해 세 가지 알고리즘의 성능을 비교 분석하게 되었고 실행 시간을 비교할 수 있게 되었다.

1. 실행 화면

**Task #1**

주어진 그래프인 64P의 그래프를 통하여 우선적으로 알고리즘을 검증해보자.

도표, 라인, 원이(가) 표시된 사진

자동 생성된 설명

주어진 그래프의 모습이다.

0-7의 최단 경로는 0-2-4-7로 12의 비용

0-5의 최단 경로는 0-3-5로 19의 비용을 지니게 된다.

다익스트라의 경우에는 모든 거리의 최소비용을 확인가능하므로 3가지 알고리즘을 테스트하여 확인해보자.  
텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

위의 결과처럼 다익스트라 알고리즘은 모든 경로에 대한 최단경로를 출력함을 확인하였고 예상했던 최단경로와 비용이 동일하게 나타났다.

A\*알고리즘, 유전자알고리즘도 동일한 결과를 나타냄을 확인할 수 있다. TASK1을 통하여 주어진 1,2,3 문제에 대한 해를 낼 수 있었으며 이를 기반으로 5번 문제를 해결할 발판을 마련할 수 있었다.

**Task #2**

앞선 코드 분석에서와 동일하게 고정된 랜덤시드 5개를 이용하여서 5개의 그래프를 생성하였다.

주어진 조건에 부합하도록 노드는 1300개 간선은 3000개를 생성하였으며

0에서 50, 0에서 900, 0에서 400을 진행하였다.  
텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 전자제품, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

각 그래프 별로 시간결과가 나온모습을 확인할 수 있다.

테스트를 진행하는 도중 유전알고리즘의 파라미터 조절에 따라 결과에 약간씩 오류가 있거나 안나오는 모습을 확인할 수 있었다. 이는 위 단점부분에 해당하는 문제점으로 확인된다. Local minma에 빠지거나 파라미터조절의 부족함, 부재로 판단이 된다. 따라서 중간중간 파라미터를 조절해주며 진행하였다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

이 파라미터가 가장 안정적인 결과를 보임을 확인할 수 있었다. 테스트 과정에서 코드를 수정하였다.

이후 위의 결과를 파이썬 코드를 사용하여 그래프화 시켜보았다.

텍스트, 라인, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 라인, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명

유전자알고리즘의 값이 커 로그스케일로 코드를 작성하였다. 위에서부터 0-50, 0-900, 0-400이다.

1. 결론

1. Dijkstra 알고리즘

0에서 50: 대부분의 그래프에서 일정한 실행 시간을 보이고 비교적 안정적이다. 실행 시간은 대체로 4000~6000 ms 사이에 분포한다.

0에서 900: 0에서 50으로 가는 실행 시간과 유사하게 일정한 범위 내에서 존재한다. 실행 시간은 대략 4000~9000 ms 사이에 분포한다.

0에서 400: 일정한 실행 시간을 보이며, 기존 데이터와 유사한 패턴을 유지한다.

2. A 알고리즘\*

0에서 50: Dijkstra 알고리즘에 비해 조금 더 변동이 크다. 대체로 2000~5000 ms 사이에 분포한다.

0에서 900: Dijkstra와 유사하게 일정한 범위 내에서 유지된다. 실행 시간은 2000~14000 ms 사이에 분포한다.

0에서 400: 기존 데이터와 유사한 실행 시간 패턴이다.

3. 유전 알고리즘

0에서 50: 다른 알고리즘에 비해 실행 시간이 매우 길다. 특히 특정 그래프에서 큰 변동을 보인다. 실행 시간은 71만~2.96천만 ms 사이에 매우 넓게 분포한다.

0에서 900: 여전히 다른 알고리즘에 비해 긴 실행 시간을 보인다. 실행 시간은 대략 1.03천만~1.69천만 ms 사이에 분포한다.

0에서 400: 실행 시간은 대략 2.63천만~3.41천만 ms 사이에 분포한다.

총 정리

Dijkstra와 A 알고리즘은 비교적 일정하고 안정적인 실행 시간을 보인다. 특히 작은 범위에서 큰 변동이 없음을 확인할 수 있다. A\* 알고리즘이 Dijkstra 알고리즘보다 약간 더 빠른 경향을 보인다.

유전 알고리즘은 실행 시간이 매우 길고 그래프의 크기와 상관없이 큰 변동을 보인다. 이는 유전 알고리즘이 더 복잡한 최적화 문제를 해결하는 데 적합하지만 실행 시간 및 정확도에서는 떨어짐을 확인할 수 있었다. 이의 원인은 아까 분석했던 단점으로 판단된다.

따라서 실시간 경로 탐색이나 빠른 응답이 필요한 경우에는 Dijkstra 또는 A\* 알고리즘을 사용하는 것이 더 적절하며, 복잡한 최적화 문제를 해결할 때는 유전 알고리즘을 고려후에 선택하면 될 것이다.