효율적인 택배 배달 알고리즘

* 중간 보고서 -

**2013104042 2013104087**

**1조 강명훈 신선경**

목차

1. 내용 분석

2. 문제 해결 방법

3. 알고리즘 구현

4. UI 구현

5. 변경 및 추가 사항

**1. 내용 분석**

주어진 문제의 목표

* 주어진 경로와 배달 목록이 존재 할 때 최소한의 순환으로서 배달을 완료 할 수 있는 알고리즘을 구현 할 것

목표를 달성 하기 위해 분석한 조건

* 경로는 이미 정해 져있고 택배가 수송 할 수 있는 N값은 랜덤으로 주어진다. 따라서 N값이 어떠한 값이 주어지더라도 성립 할 수 있는 알고리즘이 필요하다.
* 최소한의 순환을 돌기 위해서는 주어진 경로를 한번 순환 할 때 최대한 많은 택배를 수송을 함으로서 빠르게 배달 목록의 값들을 0으로 만드는 것이 중요하다.
* 위와 같은 문제를 해결 하기 위해서는 반드시 택배 배달 차량이 비어있으면 안되며 항상 가득 찬 상태로 배달을 실행하여 최대한 많은 배달을 실행해야 한다.

문제 해결을 위한 정보 가공

자료 테이블

가로 측: 배달 해야 하는 장소

세로 측: 배달 물이 저장 되어 있는 장소

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I | J |
| A | 0 | 2 | 15 | 8 | 16 | 8 | 12 | 1 | 20 | 2 |
| B | 2 | 0 | 16 | 18 | 20 | 10 | 18 | 9 | 15 | 19 |
| C | 3 | 9 | 0 | 5 | 10 | 12 | 8 | 5 | 3 | 20 |
| D | 10 | 11 | 3 | 0 | 15 | 9 | 20 | 6 | 10 | 15 |
| E | 5 | 10 | 2 | 6 | 0 | 6 | 11 | 7 | 15 | 8 |
| F | 3 | 6 | 15 | 15 | 1 | 0 | 6 | 5 | 4 | 7 |
| G | 11 | 1 | 6 | 18 | 4 | 8 | 0 | 0 | 3 | 10 |
| H | 5 | 8 | 8 | 4 | 15 | 7 | 10 | 0 | 2 | 6 |
| I | 8 | 5 | 19 | 5 | 7 | 6 | 15 | 2 | 0 | 3 |
| J | 3 | 2 | 20 | 1 | 18 | 3 | 4 | 3 | 8 | 0 |

**2. 문제 해결 방법**

**2.1. 문제 해결 방법 설명**

문제 해결을 위해 알고리즘 2가지를 사용한다.

**첫 번째 알고리즘**

다음 장소의 배달 물을 우선적으로 생각하는 알고리즘

알고리즘 구현 방법

Flow Chart



알고리즘 설명

* N값을 입력 받음
* A 출발 한다고 가정한다.
* 만약 N 값이 A->B로 배달해야 하는 물품의 개수보다 크면 택배 차에 A->B 배달 물품을 집어넣는다.
* 집어 넣은 후 N값이 0보다 큰지 판단을 하고 크다면 배달 차에 A->C로 배달해야 할 물품을 집어 넣는다.
* A ->C로 배달 하는 물품이 남은 공간 보다 크다면, 남은 공간 의 크기만큼 A->C로 배달 해야 하는 물품을 집어 넣고 배달 차를 B로 옮긴다.
* 만약 A->C로 배달 해야 하는 물품도 다 들어 간다면 A->D로 배달 해야 하는 물품 도 집어 넣는다.
* 만약 N 값이 A->B로 배달하는 물품의 개수보다 작다면 최대로 실을 수 있는 물품을 싣고 이동을 한다. 그 다음 B노드에서는 배달 차에 있는 물품을 모두 내리고 다시 B->C로가는 물품의 수와 비교를 한다.
* 이러한 알고리즘을 모든 장소들을 방문 할 때마다 적용 시킨다.
* 이러한 알고리즘을 적용 후에 아직 배달을 하지 않은 영역이 존재 하게 되는데 배달 목록에서 0으로 이루어진 대각선 아래 영역이다.
* 이러한 영역을 배달 하기 위해 이 때 알고리즘은 위에 적용 할 수 없기에 새로운 알고리즘을 대입 한다.
* 알고리즘 배달 해야 하는 우선 순위를 B, C, D … J 순으로 우선 순위를 설정하고 택배 차의 max size를 만족 할 때 까지 B, C, D 순으로 물건을 택배 차에 실어 모든 영역이 0으로 되도록 알고리즘을 계속 반복한다.
* J를 지나고 다시 A로 이동한다면 이 때 사이클을 돌았다고 판단하고 count를 1 증가 시킨다.
* 이 때 J를 지날 경우 다음 장소로서 A를 지정 해놓았기 때문에 위 알고리즘을 A에 다시 적용 시킨다.
* 모든 영역이 0인 것을 확인하기 위해 각각의 삼각형 영역에 있는 모든 맨 끝의 장소의 물품이 0이 되면 알고리즘을 종료한다.
* 종료 후 사이클을 돈 횟수를 출력한다.
* Input 값은 택배 배달 차의 용량이며 Output은 얼마나 돌았는지 Cycle의 횟수이다.

**두 번째 알고리즘**

최소 배달 물을 가진 장소를 우선 순위로 선택하는 알고리즘

Flow Chart



* N값을 입력 받는다.
* A에서 처음 출발한다고 가정
* A에서 다음 장소들에게 배달해야 할 물품을 모두 탐색하여 최소 배달 양을 가진 장소를 알아 낸다.
* 탐색 후 가장 최소의 배달 량을 가진 장소의 물품부터 택배 차에 싣는다.
* 이 때 최소의 배달 량은 현재 장소의 다음 장소들의 값에서 실행을 해야 한다.
* 이 때 N값이 최소 배달 양보다 크다면 그 다음 최소 배달 양도 택배 차에 싣는다. 이 때 택배 차의 용량이 한계가 된다면 택배 차를 다음 장소로 이동 시킨다.
* 다음 장소에서도 택배 차가 가득 차지 않을 경우 이와 같은 방법을 계속 사용한다.
* 만약 N 값이 최소 배달 양보다 크기가 작다면 최소 배달 양에서 N크기만큼 택배를 가져와 이동 시킨다.
* 이러한 알고리즘을 적용 후에 아직 배달을 하지 않은 영역이 존재 하게 되는데 배달 목록에서 0으로 이루어진 대각선 아래 영역이다.
* 이러한 영역을 배달 하기 위해 이 때 알고리즘은 위에 적용 할 수 없기에 새로운 알고리즘을 대입 한다.
* 알고리즘 배달 해야 하는 우선 순위를 B, C, D … J 순으로 우선 순위를 설정하고 택배 차의 max size를 만족 할 때 까지 B, C, D 순으로 물건을 택배 차에 실어 모든 영역이 0으로 되도록 알고리즘을 계속 반복한다.
* J를 지나고 다시 A로 이동한다면 이 때 사이클을 돌았다고 판단하고 count를 1 증가 시킨다.
* 모든 영역이 0인 것을 확인하기 위해 각각의 삼각형 영역에 있는 모든 맨 끝의 장소의 물품이 0이 되면 알고리즘을 종료한다.
* J를 지나고 다시 A로 이동한다면 이 때 사이클을 돌았다고 판단하고 count를 1증가 시킨다.
* 이 때 J를 지날 경우 다음 장소로서 A를 지정 해놓았기 때문에 위 알고리즘을 A에 다시 적용 시킨다.
* 모든 영역이 0인 것을 확인하기 위해 각각의 삼각형 영역에 있는 모든 맨 끝의 장소의 물품이 0이 되면 알고리즘을 종료한다.
* 종료 후 사이클을 돈 횟수를 출력한다.
* Input 값은 택배 배달 차의 용량이며 Output은 얼마나 돌았는지 Cycle의 횟수이다.

**2.2 문제 해결 방법의 특징**

알고리즘들의 장점

* 택배 차에 물건을 실을 시에 무조건 최우선 우선순위를 다음 지점의 장소 배달 물로 설정을 하기 때문에 택배 차가 도중에 빈 공간을 가지고 이동하는 경우가 발생 하지 않도록 설계 하였다. 따라서 순환 시에 최대한 많은 배달 할 수 있는 점에 초점을 맞추었다.

알고리즘의 단점

* 기존에 효율적으로 증명된 그래프 알고리즘을 사용하거나 변형시키는 것이 아닌 개인이 만들어낸 알고리즘이기 때문에 기존 알고리즘과 효율성 비교가 어렵다. 따라서 제시한 두 가지 알고리즘 간의 효율성만 비교가 가능하다.
* 주어진 알고리즘은 2가지 방법이 각각 합쳐진 방식의 알고리즘이다. 자료의 삼각형을 대각 선 기준으로 위와 아래 배달 하는 방식을 다르게 해야 하기 때문이다. 자료의 대각선 위 삼각형을 배달하는 방식은 각각의 알고리즘 마다 다르나 아래의 삼각형을 배달하는 방법은 동일하게 만들었다. 문제는 이러한 아래에 있는 자료를 검색하는 알고리즘은 최우선 순위를 가장 먼저 방문해야 하는 장소로 설정을 해놓고 배달 차가 모두 찰 경우 바로 한 바퀴를 돌아야 하기 때문에 순환을 많이 해야 할 수도 있다.

알고리즘의 특징

* 장점에서도 이야기 했지만 한번의 순회에서 최대한 많은 배달을 하기 위한 방법을 생각하였고 그러한 방법으로 생각해낸 방법이 배달 도중 택배 차에 빈 공간이 발생하지 않게 하여 최대한 많은 물류의 이동을 할 수 있게 하는 것이었다. 이러한 알고리즘 구현을 통해 1순회당 최대한 많은 배달을 실행 할 수 있게 설계를 해보았다.

**3. 알고리즘 구현**

* Input 값은 택배 차의 용량 N 값을 받은 후 알고리즘을 선택할 수 있게 합니다.
* Output 값으로는 한 번의 순환마다 순환 횟수를 증가시키고 변화된 데이터들을 출력할 수 있게 한다.
* 첫 번째 알고리즘을 구현하고 있으며 기본적인 사용 되는 객체와 함수 기능의 틀은 완료가 된 상태입니다. 현재 함수의 기능을 짜고 있으며 비 효율적인 패턴이나 알고리즘을 수정하고 있는 과정입니다. 또한 첫 번째 알고리즘 구현 도중 현재 A에 저장된 배달 물들을 다음 배달 물로 전달하는 것은 문제가 없으나 그 이후 B부터 J까지 장소에서 그 전의 장소에 배달하는 점에 있어 문제가 발생하여 그것을 수정 하였습니다.
* 현재의 알고리즘이 일반화가 되어있는 점이 적어서 if문과 for문의 반복되는 양이 많다는 문제가 있습니다. 이러한 문제를 해결 하기 위해 알고리즘을 더 간결화 하려고 생각하고 있습니다.

**4. UI 구현**



**5. 변경 및 추가 사항**

변경 된 점

1차 발표 시에는 Dijkstra 알고리즘을 이용하여 그러한 알고리즘의 비효율적인 측면을 예외처리로서 해결해 나아가면서 좀 더 효율 적인 알고리즘을 제작하는 것이 목적이었다. 하지만 실제로 알고리즘을 그래프 이론을 기반으로서 문제에 적용을 시키고자 하였으나 매우 많은 불편함이 존재함을 알게 되었다. 따라서 기존의 그래프 이론이 아닌 우리만의 새로운 방식의 알고리즘을 제작하여 문제를 해결 하고자 하는 것으로 방향을 변경하였다. 우리만의 알고리즘을 사용하기 때문에 효율성 비교는 우리가 제시한 알고리즘끼리 비교를 할 것이다.