|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Report**   |  |  |  | | --- | --- | --- | | **제목: 정보시스템기술 Homework #2**  **<과제물 제출 전 체크리스트>** | | | | **1** | **이 과제물은 내가/우리가 직접 연구하고 작성한 것이다.** | **■** | | **2** | **인용한 모든 자료(책, 논문, 인터넷자료 등)의 인용표시를 바르게 하였다.** | **■** | | **3** | **인용한 자료의 표현이나 내용을 왜곡하지 않았다.** | **■** | | **4** | **정확한 출처 제시 없이 다른 사람의 글이나 아이디어를 가져오지 않았다.** | **■** | | **5** | **정확한 출처 제시 없이 여러 사람의 글이나 아이디어를 짜깁기하지 않았다.** | **■** | | **6** | **과제물 작성 중 도표나 데이터를 조작(위조 또는 변조)하지 않았다.** | **■** | | **7** | **과제물은 다른 사람으로부터 받거나 구매하여 제출하지 않았다.** | **■** | | **8** | **이 과제물에 실질적으로 참여하지 않은 사람을 공동 제출자로 명기하지 않았다.** | **■** | | **9** | **이 과제물과 동일한 내용을 다른 교과목의 과제물로 제출한 적이 없다.** | **■** |  |  |  | | --- | --- | | **과목명** | **정보시스템기술 (IMEN281)** | | **담당교수** | **김병인** | | **학 과** | **산업경영공학과** | | **학 번** | **20222027** | | **성 명** | **명우성** | | **제출일** | **2022.11.10** | |
| 1. **목적**   **TSP problem을 다양한 방법으로 풀어가면서, Greedy solution인 Nearest Neighbor Solution을 Node exchange을 통해 optimal에 가깝게 근사해본다. 이 과정에서 알고리즘 수행시간을 줄이기 위해 노력하며 다이나믹 프로그래밍 방식을 익힌다. 또한 수업시간에 배웠던 예외처리, 파일 입출력, Class 디자인, 시각화 등을 해보며 프로그래밍을 통한 문제 해결 능력을 향상시킨다.**  **2. 실험방법과 알고리즘**  **Full Enumeration(Permutation)**    **Node id를 key, (위도,경도)를 value로 하는 딕셔너리를 만들어 nodelist에 입력시켰다. 두 노드 사이의 거리를 나타내는 Adjacency Matrix를 만들었다. 노드들 리스트를 permutation으로 만들고 루프를 돌려 두 노드사이의 거리를 계속구했고 total distance에 더해 총 total distance를 만들었다. 이 후 total distance가 최소 거리이면 업데이트하고, 투어의 리스트도 업데이트 하는 방식으로 최적해를 구했다. Full enumeration이기 때문에 최적해가 구해진다.**  **Nearest Neighbor**    **기본적으로 Adjacency Matrix의 element aij가 i노드와 j노드 사이의 거리 이므로, 행마다의 argmin과 min값이 가장 가까운 거리에 있는 노드 그리고 그 노드까지의 거리를 의미한다. 이 알고리즘을 수행하기 위해 먼저 필요없는 0번째행과 0번째열, 그리고 diagonal element는 inf값으로 초기화했다. 또 방문한 노드는 재방문 하면 안되므로 방문한 노드를 의미하는 열은 inf값을 넣었다. 이렇게 구한 Nearest Neighbor는 Greedy solution으로 optimal solution은 아니지만 매우 빠른 알고리즘이다.**  **Nearest Neighbor + Greedy 2 Opt**    **기존의 path와 새로운 path사이의 차이를 a라고하면 a = dis(i->i+1) + dis(j->j+1) – dis(i-1->j) – dis(i->j+1)이다. 이 거리가 양수이면 node exchange가 의미가 있다는 의미이므로, 즉시 투어리스트를 변경하고 루프를 다시 돈다.**  **Nearest Neighbor + Greedy 2 Opt**    **Greedy 2 opt와 다른점은 즉시 node exchange를 하는것이 아니라 루프를 다 돌아보고 가장 exchange의 효과가 큰 node와 exchange하는것이다. 이를 위해 후보리스트라고 하는 리스트를 만들고 노드 exchange의 효과가 있는 노드들은 전부 리스트에 담아 놓은 후, 루프를 돌고 난 후 그 거리들을 비교해서 가장 효과가 좋은 노드와 교환한다.**  **3. 결과 토의**  **맥북 m1칩 (CPU8코어 GPU14코어)로 실험 진행 결과**   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | **Permutation** | **Nearset N** | **Greedy 2 opt** | **Full2 Opt** | | **N = 10** | **1039msec** | **0.13msec** | **0.17msec** | **0.17msec** | | **N= 20** | **-** | **0.33msec** | **0.49msec** | **0.5msec** | | **N =100** | **-** | **8.519msec** | **10.59msec** | **15.48msec** | | **N= 1000** | **-** | **574.59msec** | **977msec** | **1028msec** | | **N = 10000** | **-** | **59106msec** | **100301msec** | **99583msec** |   **Permutation을 이용해서 Full Enumeration을 구하는 방식은 확실히 optimal solution을 구할 수 있지만 너무 오랜 시간이 걸렸다. Nearest Neighbor에서 Greedy 2 OPT, Full2 OPT로 갈수록 소요시간은 늘었지만 solution quality(total distance)가 줄었다.**  **4. 실행 화면 스크린샷**  **N = 10 Case**   |  |  | | --- | --- | |  |  | |  |  |     **5. 타인의 도움**  **다른 도움은 받지 않았습니다.**  **6. 과제에 소요된 시간 기록**  **수업시간 외에 24시간 소요** |