ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÀI TẬP MÔN HỌC
PHÂN TÍCH THIẾT KẾ THUẬT TOÁN

Sinh viên: Đỗ Phương Duy - 23520362

Sinh viên: Nguyễn Nguyên Khang - 22520623

Ngày 21 tháng 11 năm 2024

Mục lục

1 Bài 1: Bài toán Set Cover: 3
1.1 Cách giải thứ nhất . . . 3
1.2 Cách giải thứ hai . . . 4

2 Travelling Salesman Problem (TSP) 4
2.1 Problem Description . . . 4
2.2 Formal Definition . . . 4
2.3 Approximation Algorithms . . . 5
2.3.1 Method 1: Greedy Algorithm . . . 5
2.3.2 Method 2: 2-Approximation Algorithm (Minimum Spanning Tree - MST) 5

1 Bài 1: Bài toán Set Cover:

1.1 Cách giải thứ nhất

Chi tiết cách giải:

- **Khởi tạo:**
  - Mảng kết quả là một tập rỗng result $= \emptyset$
  - Sắp xếp các tập con S_i theo kích thước giảm dần
- **Lặp:**
  - **Bước 1:** Chọn S[0], tập con có kích thước lớn nhất còn lại
  - **Bước 2:** Với mỗi phần tử trong S[0], nếu phần tử đó chưa nằm trong result, thêm nó vào result.
  - **Bước 3:** Loại bỏ tập S[0] khỏi danh sách S.
  - **Bước 4:** Tìm tất cả các phần tử U chưa được bao phủ bởi result.
  - **Bước 5:** Với mỗi tập S_i, đếm số lượng phần tử U chưa bao phủ mà nó chứa. Loại bỏ S_i nào không chứa bất kỳ phần tử nào trong các phần tử U còn lại.
  - **Bước 6:** Sắp xếp lại danh sách S theo số lượng phần tử U mà chúng chứa, giảm dần
- **Kết thúc:**
  - Lặp lại cho đến khi tất cả các phần tử U được bao phủ
  - Kết quả là danh sách các tập con S_i trong result.

Code Python:

Listing 1: Thuật toán Greedy giải bài toán Set Cover

```python
def greedy_set_cover(U, S):
    result = []
    covered = set()

    S = sorted(S, key=lambda x: len(x), reverse=True)

    while covered != set(U):
        best_set = S[0]
        result.append(best_set)

        covered.update(best_set)

        S = S[1:]

        uncovered = set(U) - covered

        S = [s for s in S if any(u in s for u in uncovered)]

        S.sort(key=lambda x: len(set(x) & uncovered), reverse=True)

    return result
```

1.2 Cách giải thứ hai

Chi tiết cách giải:

- **Khởi tạo:**
  - Khởi tạo một tập rỗng result để lưu các tập đã chọn
  - Tạo một tập covered để lưu các phần tử đã được bao phủ
  - Duyệt qua các tập S_i tuần tự theo thứ tự cho trước
- **Lặp:**
  - Nếu S_i chứa ít nhất một phần tử chưa được bao phủ, thêm S_i vào result và cập nhật tập covered.
  - Nếu tất cả các phần tử trong U đã được bao phủ, dừng vòng lặp
- **Kết thúc:**
  - Trả về danh sách các tập con S_i đã chọn

Code Python:

Listing 2: Thuật toán Lựa chọn ngẫu nhiên giải bài toán Set Cover

```python
def simple_set_cover(U, S):
    result = []
    covered = set()

    for s in S:
        if not set(s).issubset(covered):
            result.append(s)
            covered.update(s)

        if covered == set(U):
            break

    return result
```

2 Travelling Salesman Problem (TSP)

2.1 Problem Description

The Travelling Salesman Problem (TSP) aims to find a Hamiltonian cycle in a graph such that:

- Each vertex is visited exactly once.
- The total cost of the cycle is minimized.

2.2 Formal Definition

Given a graph $G = (V, E)$:

- $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices.
- E is the set of edges with weights $c(v_i, v_j)$ representing cost, time, or distance.

Find a Hamiltonian cycle such that the total weight is minimized.

2.3 Approximation Algorithms

2.3.1 Method 1: Greedy Algorithm

Idea: Start from an arbitrary vertex. At each step, choose the smallest edge connecting the current vertex to an unvisited vertex.

Algorithm:

1. Select a starting vertex v_1.
2. While there are unvisited vertices:
    - Choose the smallest edge (v_i, v_j) where v_j is unvisited.
3. Return to the starting vertex to complete the cycle.

Complexity: $O(n^2)$, where $n = |V|$.

2.3.2 Method 2: 2-Approximation Algorithm (Minimum Spanning Tree - MST)

Algorithm:

1. Construct the MST of G using Kruskal's or Prim's algorithm.
2. Perform an Euler Tour on the MST to visit all vertices.
3. Remove duplicate visits, keeping the first occurrence of each vertex.

Complexity: $O(n^2)$.
Approximation Ratio: The cycle's total weight is at most twice the optimal solution.
Idea: Build a Minimum Spanning Tree (MST) and perform an Euler Tour to construct a Hamiltonian cycle.

Tài liệu