Homework 6: Greedy Methods

Bùi Khánh Duy - 20001898

Lựa chọn 1: Thuật toán tự tìm hiểu và trình bày

Phần 1: Thuật toán Dijkstra

Phần code sẽ nằm trong file `dijkstra.cpp`

Exercise 9.3

1. (2) **Explain what adjustments if any need to be made in Dijkstra's algorithm and/or in an underlying graph to solve the following problems.**
    a. **Solve the single-source shortest-paths problem for directed weighted graphs.**

        Các bước như sau:

        1. Khởi tạo một mảng khoảng cách và một mảng kiểm tra đã đi qua chưa cho tất cả các đỉnh trong đồ thị. Đặt khoảng cách của đỉnh nguồn thành 0 và tất cả các đỉnh khác thành vô cùng. Đặt tất cả các đỉnh thành chưa đi qua.
        2. Chọn đỉnh có khoảng cách tối thiểu từ nguồn. Đỉnh này sẽ là đỉnh hiện tại.
        3. Đối với mỗi cạnh đi ra từ đỉnh hiện tại, tính khoảng cách đến đỉnh đích. Nếu khoảng cách nhỏ hơn khoảng cách hiện tại được lưu trong mảng khoảng cách cho đỉnh đích, cập nhật mảng khoảng cách với khoảng cách mới.
        4. Đánh dấu đỉnh hiện tại là đã đi qua.
        5. Lặp lại các bước 2-4 cho đến khi tất cả các đỉnh đã được truy cập.
        6. Mảng khoảng cách thu được sẽ chứa khoảng cách ngắn nhất từ đỉnh nguồn đến mọi đỉnh khác trong đồ thị

    b. **Find a shortest path between two given vertices of a weighted graph or digraph. (This variation is called the *single-pair shortest-path problem*.)**

        Thay vì để thuật toán chạy cho đến khi đi qua hết tất cả các đỉnh, chỉ cần đến đích là ngừng lại. Điều này giúp tiết kiệm thời gian và giảm độ phức tạp.

    c. **Find the shortest paths to a given vertex from each other vertex of a weighted graph or digraph. (This variation is called the *single-destination shortest-paths problem*.)**

        Coi đỉnh đích như đỉnh nguồn và giải với bài toán Dijkstra thông thường.

d. **Solve the single-source shortest-paths problem in a graph with non negative numbers assigned to its vertices (and the length of a path defined as the sum of the vertex numbers on the path).**

   Ta coi giá trị ở đỉnh thành trọng số của cạnh ⇒ Độ dài đường đi = tổng giá trị của các đỉnh trên nó.

5. (5) **Write pseudocode for a simpler version of Dijkstra's algorithm that finds only the distances (i.e., the lengths of shortest paths but not shortest paths themselves) from a given vertex to all other vertices of a graph represented by its weight matrix.**

```
Dijkstra(G, W, s)
  //initialize distances to infinity, except for the starting vertex
  dist = [infinity, infinity, ..., infinity]
  dist[s] = 0

  //initialize a set of unvisited vertices
  Q = {0, 1, 2, ..., n-1}

  while Q is not empty
    //find the unvisited vertex with the smallest tentative distance
    u = vertex in Q with smallest dist[u]

    //update the distances of the neighbors of u
    for each neighbor v of u
      alt = dist[u] + W[u][v]
      if alt < dist[v]
        dist[v] = alt

    //remove u from Q
    remove u from Q

  //return the distances
  return dist
```

6. (6) **Give a real life application of Dijkstra's algorithm.**

   - IP routing to find Open shortest Path First: Thuật toán Dijkstra được sử dụng rộng rãi trong các giao thức định tuyến được yêu cầu bởi các bộ định tuyến để cập nhật bảng chuyển tiếp của chúng. Thuật toán cung cấp đường đi có chi phí ngắn nhất từ bộ định tuyến nguồn đến các bộ định tuyến khác trong mạng.

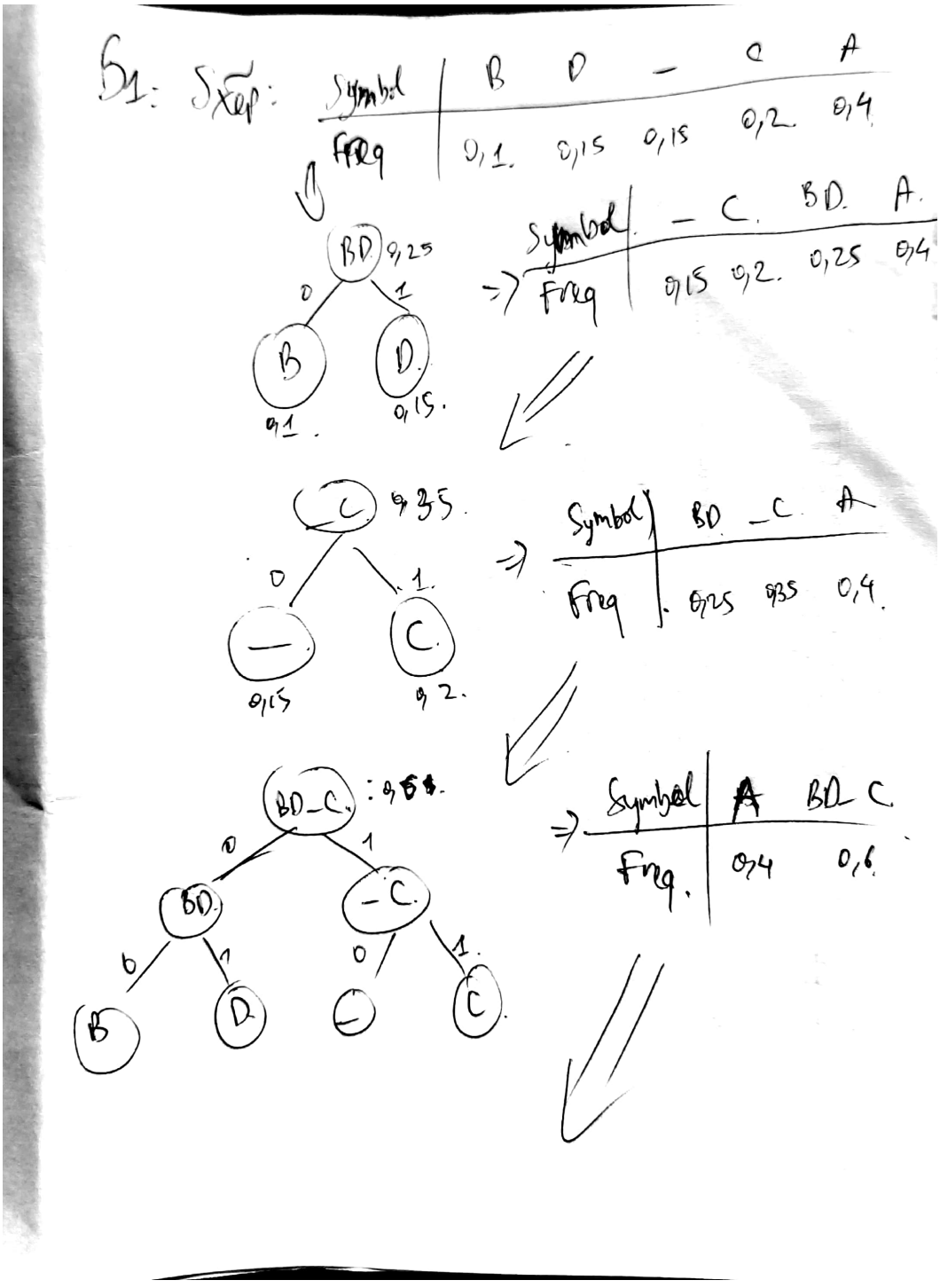
Phần 2: Cây và mã Huffman

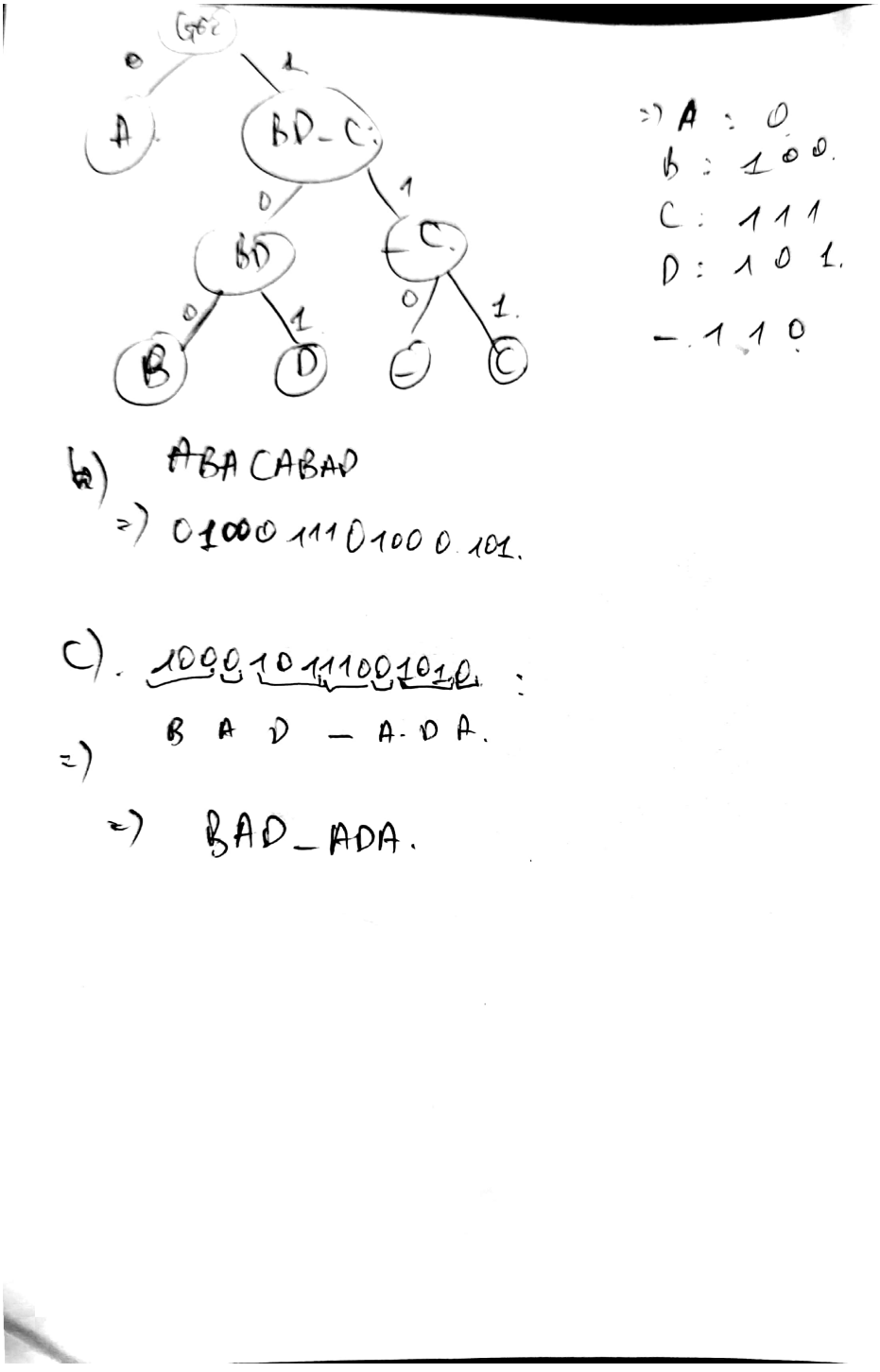
Phần code sẽ nằm trong file `huffman.ipynb`

Exercise 9.4

1. (1)
    a. Construct a Huffman code for the following data:

Symbol	A	B	C	D	_
Frequency	0.4	0.1	0.2	0.15	0.15


Bước 1: Sắp:
Symbol | B D _ C A
Freq | 0,1 0,15 0,15 0,2 0,4
BD 0,25
0 1
B D
0,1 0,15
Symbol | _ C BD A
Freq | 0,15 0,2 0,25 0,4
C 0,35
0 1
_ C
0,15 0,2
Symbol | BD _C A
Freq | 0,25 0,35 0,4
BD_C : 0,6
0 1
BD _C
0 1 0 1
B D _ C
Symbol | A BD_C
Freq | 0,4 0,6



2. (3)
    a. Đúng.
    b. Đúng
3. (8) **Give the application and features of Huffman’s encoding. Explain them briefly.**

- Conventional compression formats like PKZIP, GZIP, etc. [1]
- Multimedia codecs like JPEG, PNG, and MP3 use Huffman encoding (to be more precise the prefix codes) [1]
- Lossless image compression, where Huffman coding can be used to compress images smaller than 8 bytes/point [4]
- Text compression, where Huffman coding can be used to compress English text to a reasonable degree since the probabilities of each letter are surprisingly similar across different English texts [4]
- Audio compression, where the frequencies of audio data tend to be similar if they are adjacent to one another, so by subtracting off the previous value in time from each frequency recording, the data becomes easier to encode and has less structure [4].

Tham khảo

https://www.baeldung.com/cs/kruskals-vs-prims-algorithm

https://www.baeldung.com/cs/prim-dijkstra-difference

https://medium.com/@mitanshupbhoot/comparative-applications-of-prims-and-kruskal-s-algorithm-in-real-life-scenarios-4aa0f92c7abc