

# Đề thi cuối kỳ thực hành

## CS104 – Cấu trúc dữ liệu và giải thuật

### Bài tập 1 : Ứng dụng nén tập tin sử dụng phương pháp nén Huffman tĩnh (9 điểm)

Trong lĩnh vực công nghệ thông tin, khái niệm nén dữ liệu đề cập đến phương pháp mã hóa dữ liệu số nhằm giảm dung lượng lưu trữ so với cách thể hiện ở dữ liệu gốc. Việc nén dữ liệu ngày càng trở nên cần thiết vì không những tiết kiệm dung lượng lưu trữ mà còn giúp thuận tiện trong việc sao chép, gửi đính kèm hay chia sẻ qua mạng Internet. Đa phần các phương pháp nén đều xử lý trên Bit. Một số ứng dụng nén thường hay được sử dụng : Winzip, winrar, 7zip, ...

Có hai phương pháp nén thường được sử dụng tùy theo nhu cầu, *phương pháp nén bảo toàn dữ liệu và nén mất mát dữ liệu*. Phương pháp nén mất mát dữ liệu thường được sử dụng trên dữ liệu về đa phương tiện như ảnh, âm thanh do việc giảm chất lượng của những loại dữ liệu này không ảnh hưởng quá nhiều và có thể chấp nhận ở một mức độ cho phép. Ảnh với định dạng .jpg là một ví dụ điển hình cho việc nén mất mát dữ liệu. Cần lưu ý phương pháp nén mất mát dữ liệu không thể khôi phục được tập tin gốc như ban đầu.

Nén bảo toàn dữ liệu được xây dựng dựa trên các thuật toán tối ưu hóa việc xử lý và lưu trữ dữ liệu sử dụng thống kê và loại bỏ những phần không cần thiết hoặc lặp lại. Phương pháp nén này phù hợp với dữ liệu cần đúng với bản gốc như văn bản, yêu cầu có thể khôi phục lại tập tin gốc nguyên vẹn sau khi nén. Bài tập này hướng đến việc tìm hiểu và cài đặt phương pháp nén không mất mát dữ liệu, đó là phương pháp **nén Huffman tĩnh (Static Huffman Compression)**

Phương pháp nén Huffman tĩnh dựa trên tần số xuất hiện của các ký tự trong tập tin, từ đó biến đổi dãy bit thể hiện để lưu trữ ít hơn. Thông thường, mỗi ký tự biểu diễn dưới dạng mã ASCII đều cần 1 byte (hay 8 bit) để lưu trữ. Tuy nhiên, việc xuất hiện tất cả các ký tự trong bảng mã ASCII ít khi xảy ra nên những ký tự nào xuất hiện nhiều cần được biểu diễn ít bit hơn thay vì biểu diễn 8 bit như nhau, từ đó tiết kiệm được nhiều dung lượng lưu trữ hơn. Đây là phương pháp nén bằng **mã có độ dài thay đổi**.

Ký tự trong bảng mã ASCII	Bit để biểu diễn thông thường	Bit để biểu diễn sau khi nén
A	01000001	11
B	01000010	01
C	01000011	00
D	01000100	10

- Ví dụ, đoạn dữ liệu trong tập tin sau:

**ABCDBADA**

- Với cách lưu trữ thông thường thì cần:  $8 \text{ (ký tự)} * 8 \text{ bit} = 64 \text{ bit} = 8 \text{ byte}$  để lưu trữ.

**ABCDBADA**

**01000001 01000010 01000011 01000100 01000010 01000001 01000100 01000001**

- Với cách nén Huffman thì cần:  $8 \text{ (ký tự)} * 2 \text{ bit} = 16 \text{ bit} = 2 \text{ byte}$  để lưu trữ

**ABCDBADA**

**11010010 01111011**

Trong máy tính chỉ lưu theo byte, tức là những số bit lẻ cần phải được thêm vào để cho đủ 1 byte (8 bit). Ví dụ trên giả định rằng thông tin cần thiết cho việc giải nén được lưu trữ và xử lý riêng, nên phương pháp nén Huffman giúp **giảm đi 75% (2 byte so với 8 byte) dung lượng lưu trữ** đối với đoạn văn bản trong ví dụ. Những đoạn mã thu gọn sau khi nén được gọi là mã Huffman. Để tạo ra mã Huffman, cấu trúc dữ liệu **cây nhị phân tìm kiếm** được áp dụng, được gọi là cây Huffman.

**Các bước của thuật toán nén Huffman tính được thực hiện như sau:**

- Bước 1:** đếm số lượng xuất hiện của từng ký tự có trong văn bản, bao gồm cả các ký tự đặc biệt như ký tự xuống dòng ( $\backslash n$ ), ký tự null ( $\backslash 0$ ) hay ký tự khoảng cách.

Đoạn văn bản ví dụ: **ABCDBADA**

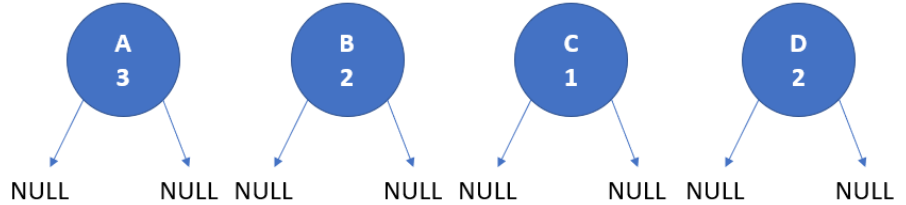
Ký tự	Tần số xuất hiện
A	3
B	2
C	1
D	2

- Bước 2:** Tiến hành dựng cây Huffman. Cây Huffman được định nghĩa là cây nhị phân tìm kiếm, trong đó quy luật duyệt cây như sau:
  - Nếu đi về hướng bên trái, mã code sẽ được thêm vào là 0
  - Nếu đi về hướng bên phải, mã code sẽ được thêm vào là 1
- Để xây dựng cây Huffman, từ bảng mã thống kê tần số xuất hiện, ta sẽ **tạo mỗi ký tự trong bảng thống kê tương ứng với một nút lá trong cây Huffman**. Sau đó, **chọn 2 nút có tần**

số xuất hiện **ít nhất** nối với nút cha. Nút cha của 2 nút này có tần số xuất hiện là tổng tần số xuất hiện của 2 ký tự ở 2 nút con. Ví dụ:

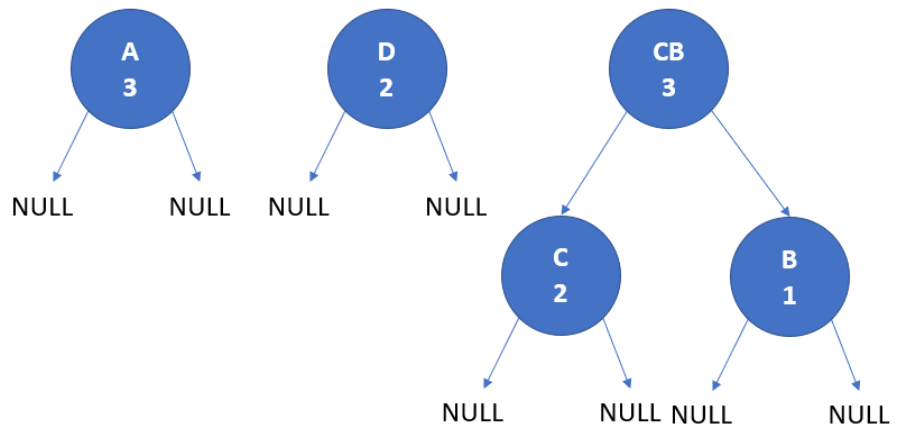
- Từ bảng mã thông kê tần số, tiến hành tạo 4 nút lá tương ứng

Ký tự	Tần số xuất hiện
A	3
B	2
C	1
D	2



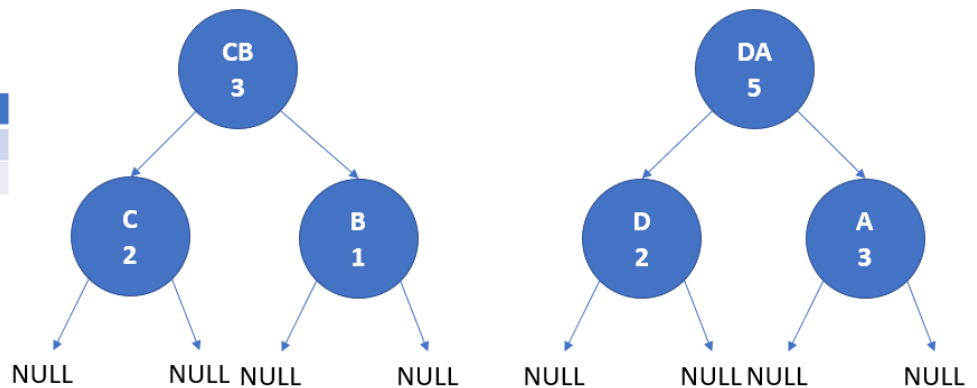
- Chọn **2 nút có tần số xuất hiện ít nhất**, tạo nút cha để nối 2 nút con, nút cha có tần số xuất hiện bằng tổng tần số xuất hiện của 2 nút con

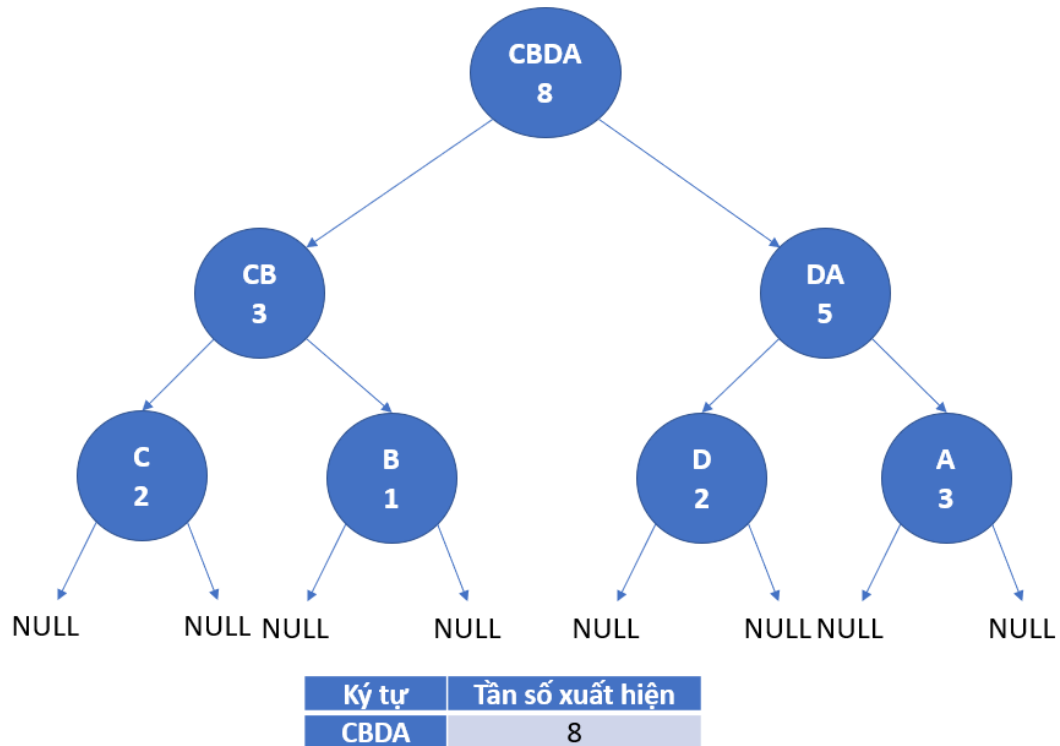
Ký tự	Tần số xuất hiện
A	3
CB	3
D	2



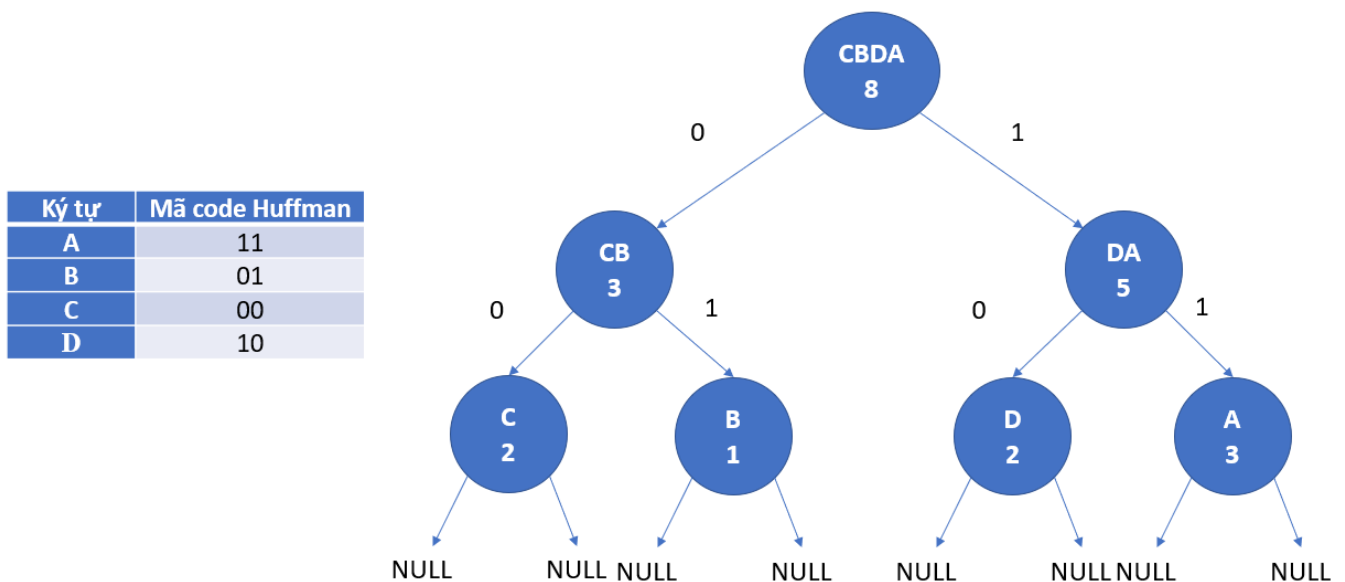
- Tiếp tục chọn 2 nút có tần số xuất hiện nhỏ nhất và tạo nút cha để nối, cho đến khi **chỉ còn 1 nút duy nhất**, đó chính là nút gốc của cây Huffman

Ký tự	Tần số xuất hiện
CB	3
DA	5





- Bước 3:** Dựa trên cây Huffman, tiến hành duyệt đến từng nút lá để đưa ra mã nén tương ứng, dựa theo quy tắc của cây Huffman như đã nêu ở bước 2:
  - Nếu đi về hướng bên trái, mã code sẽ được thêm vào là 0
  - Nếu đi về hướng bên phải, mã code sẽ được thêm vào là 1



- Bước 4:** duyệt từng ký tự trong tập tin, lưu mã bit tuần tự sao cho đủ 8 bit thì xuất ký tự đó ra tập tin nén. Do đó cần thực hiện các thao tác trên bit dữ liệu. Một số thao tác trên bit:

- Gán một bit tại vị trí thứ position là 1

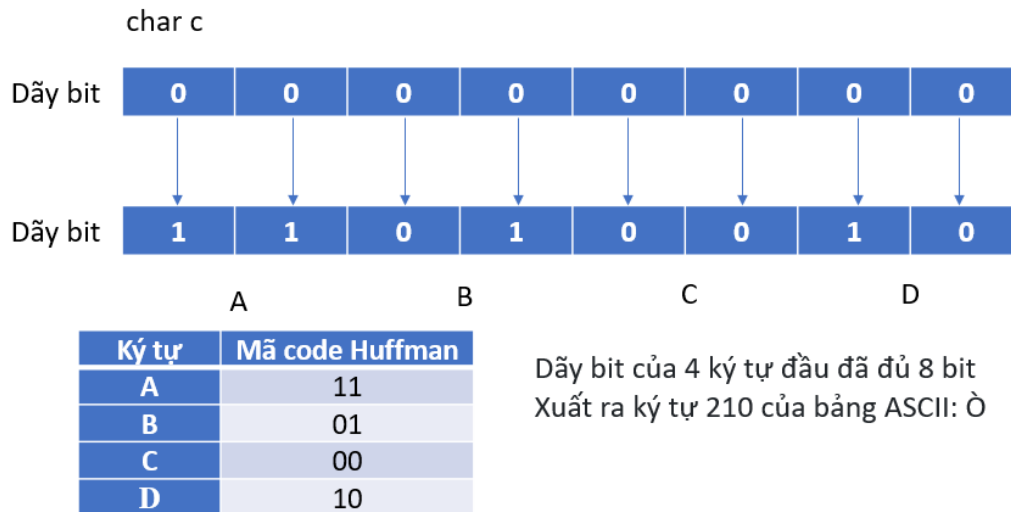
```
void OnBit(unsigned char &byte, int position)
{
    byte = byte | (1 << position);
}
```

- Gán một bit tại vị trí thứ position là 0

```
void OffBit(unsigned char &byte, int position)
{
    byte = byte & ~(1 << position);
}
```

- Vậy cần tạo một biến 1 byte, do đó kiểu char hay unsigned char là phù hợp trong ngôn ngữ C++.
- Nếu số lượng bit không đủ 1 byte, có thể tiến hành thêm dãy số 0 hay số 1 sao cho phù hợp. Sinh viên xem xét xử lý vấn đề này để tránh mất mát dữ liệu.

### ABCDBADA



**Để giải nén**, cần phải lưu trữ lại thông tin cây Huffman, thường được lưu ở đầu tập tin nén, sau đó đọc tuần tự và xử lý, xây dựng lại cây Huffman và duyệt trả từ mã nén Huffman trở về ngược lại ký tự. Gợi ý những cách lưu trữ thông tin cho việc giải nén:

- Lưu trữ bảng thống kê tần số xuất hiện kèm ký tự tương ứng:

ABCDBADA – **3A2B1C2D**

- Lưu trữ bảng mã nén Huffman kèm ký tự tương ứng:

ABCDBADA – **11A01B00C10D**

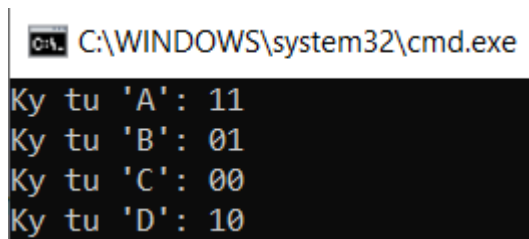
Việc lưu trữ thông tin cây Huffman có thể làm cho tập tin nén có kích thước to hơn, nhưng sẽ nhỏ dần lại nếu tập tin cần nén có dung lượng lớn, vì phần lưu trữ phục vụ giải nén này có giới hạn.

### **Yêu cầu thực hiện:**

- a. Xây dựng chương trình C++ tiến hành **nén dữ liệu** bằng phương pháp nén Huffman tĩnh theo mô tả như trên. Sinh viên **đề xuất cấu trúc dữ liệu và giải thuật** thực hiện phương pháp nén Huffman. Biết rằng mỗi nút (Node) ở cây Huffman gồm 2 con trỏ chỉ sang trái và phải, sinh viên **đề xuất cho phần lưu trữ dữ liệu** ở mỗi nút. (5 điểm)

```
struct HNode {
    //Sinh viên đề xuất phần lưu trữ dữ liệu của nút trong cây
    //...
    //Gợi ý:
    //unsigned char character; //Lưu trữ ký tự cần lưu
    //int frequency; //Lưu trữ tần số xuất hiện của ký tự
    HNode* left; //Con trỏ chỉ sang nút trái của cây
    HNode* right; //Con trỏ chỉ sang nút phải của cây
};
```

- Đầu vào gồm tên tập tin cần nén (.txt). Kết quả cho ra tập tin nén, định dạng .huf có cùng tên với tập tin cần nén. **Sinh viên chỉ thực hiện nén trên tập tin văn bản thô (.txt)**
  - Ví dụ: đầu vào gồm tập tin FileA.txt, kết quả cho ra tập tin nén FileA.huf
- **Giới hạn ký tự trong tập tin cần nén:** tất cả các ký tự trong bảng chữ cái tiếng Anh (viết hoa và viết thường), ký tự số từ 0 đến 9, ký tự khoảng trắng, ký tự xuống dòng (\n và \r), ký tự cách dòng tab (\t) và ký tự \0
- Sinh viên xuất ra mã nén của từng ký tự ra ngoài màn hình Console trong lúc nén, định dạng xuất:



```
C:\WINDOWS\system32\cmd.exe
Ky tu 'A': 11
Ky tu 'B': 01
Ky tu 'C': 00
Ky tu 'D': 10
```

- b. Xây dựng chương trình C++ thực hiện **giải nén** tập tin đã nén dựa trên phương pháp nén Huffman tĩnh. Yêu cầu tập tin sau khi giải nén phải có **độ chính xác đến 99%** so với tập tin gốc trước khi nén, phải đúng cả định dạng và vị trí. (1 điểm)
- Đầu vào gồm tên tập tin đã nén có định dạng .huf. Kết quả cho ra tập tin văn bản thô (.txt) có tên cùng với tên file nén + “Decompression”
    - Ví dụ: đầu vào gồm tập tin nén FileA.huf, kết quả cho ra tập tin văn bản FileADecompression.txt

- Tập tin nén có thể có chứa tất cả ký tự trong bảng mã ASCII, nhưng tập tin văn bản trước khi nén có giới hạn ký tự như câu a.
- c. Xây dựng chương trình C++ thực hiện **nén và giải nén tập tin văn bản tiếng Việt (.txt)** dựa trên phương pháp nén Huffman tĩnh. Yêu cầu tập tin sau khi giải nén phải có **độ chính xác đến 99%** so với tập tin gốc trước khi nén, phải đúng cả định dạng và vị trí. (1 điểm)
- Đầu vào và đầu ra quy định tương tự như câu a khi nén và câu b khi giải nén
  - **Giới hạn ký tự ở tập tin nén: không có giới hạn.**
- d. Viết báo cáo phân tích thuật toán và mã nguồn đã cài đặt ở câu a, b và nếu có ở câu c. Sinh viên cần **mô tả rõ cấu trúc và cách thức lưu trữ của tập tin nén**, đồng thời phân tích **hiệu suất nén** (dung lượng sau khi nén so với dung lượng trước khi nén), đưa ra nhận xét phương pháp nén Huffman (ưu, khuyết điểm) và đề xuất phương pháp cải tiến (nếu có) (1 + 1 điểm)
- \*Nếu phân tích tốt, sinh viên được cộng thêm tối đa 1 điểm cho bài tập này.**

## Bài tập 2 : Tìm đường đi trong mê cung (3 điểm)

Sinh viên vận dụng kiến thức đã học ở phần lý thuyết đồ thị, thực hành để giải quyết yêu cầu của bài toán tìm đường đi trong mê cung. Bài toán được mô tả như sau:

Cho một mê cung, được biểu diễn dưới dạng mảng 2 chiều với kích thước là  $M \times N$  ( $M$  là số dòng,  $N$  là số cột), mỗi một ô trong mảng thể hiện trạng thái riêng trong mê cung. Mỗi phần tử trong mảng 2 chiều có thể chứa 1 trong những giá trị sau:

- Phần tử chứa giá trị là ký tự 'S': là vị trí bắt đầu đi tìm đường trong mê cung
- Phần tử chứa giá trị là ký tự 'E': là vị trí cần đi đến
- Phần tử chứa giá trị là 0: thể hiện cho tường chắn, không có lối đi đến vị trí này
- Phần tử chứa giá trị là 1: thể hiện cho việc có đường đi đến vị trí này trong mê cung
- **Mỗi mê cung chỉ chứa duy nhất 1 ký tự 'S' và 1 ký tự 'E'**
- Ví dụ, mô tả mê cung có kích thước  $4 \times 6$  (4 dòng, 6 cột)

0	0	0	0	0	0
0	S	1	1	1	0
0	1	0	0	1	0
0	1	0	E	1	0

Thông tin của 1 mê cung được biểu diễn trong tập tin đặt tên là "Maze.txt" để cùng thư mục chứa project Visual Studio. Cấu trúc của tập tin chứa thông tin mê cung được thể hiện như sau:



- Dòng đầu tiên gồm 2 số nguyên, cách nhau bằng 1 khoảng trắng, thể hiện cho số dòng (M) và số cột (N) của ma trận mê cung
- M dòng tiếp theo, mỗi dòng gồm có N ký tự cách nhau bằng 1 khoảng trắng thể hiện cho mảng 2 chiều với giá trị được mô tả ở trên.

Maze.txt - Notepad

File Edit Format View Help

13 8

1	1	1	0	0	0	0	0
0	S	1	1	1	0	0	0
0	0	1	1	1	1	1	0
0	0	1	1	1	1	1	0
0	0	1	1	1	1	1	0
0	0	1	0	1	0	1	0
1	1	1	0	1	0	1	0
1	0	0	1	1	0	1	0
1	1	0	0	0	0	1	0
1	1	0	0	0	0	1	E
1	1	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0

## Yêu cầu bài toán:

- Viết chương trình C++ để tìm đường đi ngắn nhất từ vị trí 'S' đến vị trí 'E' trong mê cung với đầu vào là tập tin "Maze.txt" được mô tả trong bài toán. Sinh viên xuất thông tin đầu ra bao gồm mảng 2 chiều của tập tin đầu vào, bắt đầu từ vị trí 'S', nếu đi đến vị trí nào tiếp theo thì vị trí đó được gán là -1 để biểu diễn đường đi đến vị trí 'E', vị trí 'S' và 'E' cũng được thay thế bằng -1, lưu vào tập tin "Result.txt" (1 điểm).
- Viết báo cáo mô tả cấu trúc dữ liệu và giải thuật, chiến thuật sử dụng để giải quyết bài toán tìm đường đi trong mê cung, từ đó tiến hành phân tích đánh giá về chiến thuật đã sử dụng (ưu, khuyết điểm), nhận xét về độ phức tạp và khả năng tối ưu và đề xuất phương pháp cải tiến nếu có (1 + 1 điểm)

**\*Nếu phân tích tốt, sinh viên được cộng thêm tối đa 1 điểm cho bài tập này.**

- Ví dụ về đầu ra của tập tin mô tả mê cung trên:



result.txt - Notepad

File	Edit	Format	View	Help				
1		1		0	0	0	0	0
0	-1	-1	-1	-1	0	0	0	0
0	0	1	1	-1	-1	-1	0	0
0	0	1	1	1	1	-1	0	0
0	0	1	1	1	1	-1	0	0
0	0	1	0	1	0	-1	0	0
1	1	1	0	1	0	-1	0	0
1	0	0	1	1	0	-1	0	0
1	1	0	0	0	0	-1	0	0
1	1	0	0	0	0	-1	-1	0
1	1	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0

## Quy định nộp bài

- ✚ Tạo một folder đặt tên là <MSSV>, trong đó có chứa các folder con với tên sau:
  - Report: chứa báo cáo về bài làm đồ án cuối kỳ, định dạng .doc/.docx và .pdf.
  - Source Code: **không nộp toàn bộ project**, chỉ nộp các tập tin mã nguồn liên quan đến bài làm, đặt tên các tập tin theo quy tắc sau:
    - Bai1.cpp: toàn bộ mã nguồn của bài tập 1 (bao gồm cả hàm main)
    - Bai2.cpp: toàn bộ mã nguồn của bài tập 2 (bao gồm cả hàm main)
    - Các tập tin cần thiết kèm theo của cả 2 bài tập
  - References: chứa tập tin "Ref.txt" chứa thông tin tài liệu tham khảo.
- ✚ Sinh viên nén toàn bộ folder thành tập tin nén <MSSV>.rar/zip và nộp lên Moodle.
- ✚ **Thời gian làm bài: một tuần, bắt đầu từ 12h trưa thứ 4 (15/04/2020) đến 12h trưa thứ 4 tuần tiếp theo (22/04/2020).**
- ✚ Sinh viên được hỗ trợ template gợi ý báo cáo (file .doc/docx), nội dung báo cáo sinh viên tự trình bày. Font chữ là Times New Roman, kiểu chữ 13, canh lề Justify.
- ✚ Tổng điểm có thể lớn hơn 10 điểm để sinh viên có nhiều lựa chọn hơn khi thực hiện đồ án nhưng tối đa vẫn là 10 điểm.
- ✚ **Giới hạn dung lượng tập tin cần nén ở tất cả các câu trong bài 1: không quá 100MB**
- ✚ **Sinh viên không được phép sử dụng bất kỳ hàm sắp xếp nào có sẵn trong hệ thống.**
- ✚ **Sinh viên không được phép sử dụng stack và queue có sẵn trong hệ thống**
- ✚ **Tất cả những bài làm giống nhau (kể cả mã nguồn) đều bị 0 điểm.**
- ✚ **Tất cả những bài làm gây ra hiện tượng rò rỉ bộ nhớ bị trừ 50% tổng điểm.**
- ✚ **Yêu cầu cần phải viết báo cáo lại những gì đã làm, nếu không viết báo cáo, sinh viên sẽ bị trừ 1 điểm mỗi bài tập.**