Cấu Trúc Dữ Liệu Và Giải Thuật

Bài tập lớn 1

JJK RESTAURANT OPERATIONS

nhóm thảo luận CSE
https://www.facebook.com/groups/211867931379013

Tp. Hồ Chí Minh, Tháng 6/2023

Mục lục

1 Kiên thức cần có

- Danh sách liên kết đôi
- Danh sách liên kết vòng
- Sell sort

```
class customer {
public:
    string name;
    int energy;
    customer* prev;
    customer* next;
    friend class Restaurant;
public:
    customer(){}
    customer(string na, int e, customer* p, customer *ne)
    : name(na), energy(e), prev(p), next(ne){}
    ~customer(){
        delete prev;
        delete next;
    }
    void print() {
        cout << name << "-" << energy << endl;
    }
};
```

- *NAME*: một chuỗi ký tự trong bảng chữ cái Alphabet (bao gồm cả chữ viết thường và viết hoa) liên tục không có khoảng trắng, biểu thị tên của khách hàng.
- *ENERGY*: một số nguyên biểu thị năng lượng của các chú thuật sư (giá trị dương), và oán linh (giá trị âm).
- *NUM*: một số nguyên với ý nghĩa khác nhau ứng với từng lệnh xử lý khác nhau. Và ứng với mỗi lệnh thì giá trị NUM này sẽ có các khoảng giá trị khác nhau.
- *MAXSIZE*: số lượng bàn tối đa mà nhà hàng có thể phục vụ. Tuy nhiên giá trị này có thể tạm thời thay đổi trong quá trình vận hành nhà hàng.

```
private:
    customer * customerX; //! lưu trữ danh sách khách hàng đang ở trong bàn
    int sizeCustomerInDesk;
    customer * customerQueue; //! mô tả danh sách khác hàng đang trong hàng đợi chỉ sử dung ne
    int sizeCustomerQueue;
```

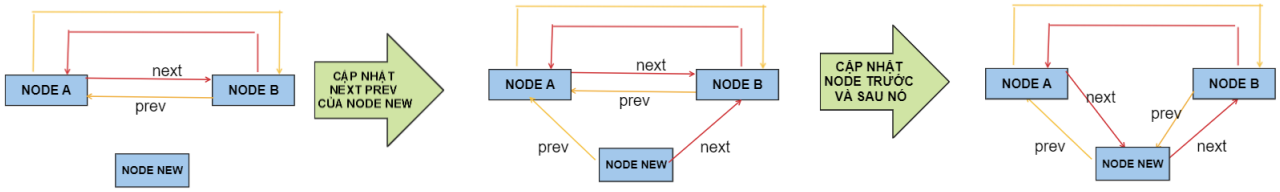
- *customerX* khách hàng được vị trí gần nhất vừa có sự thay đổi (ngồi vào bàn hoặc ra khỏi bàn). **Được biểu diễn Danh sách liên kết vòng kép Doubly Circular linked list**
- *sizeCustomerInDesk* số lượng khách hàng đang có trong bàn
- *customerQueue* danh sách khách hàng đang chờ **Được biểu diễn Danh sách liên kết vòng Circular linked list**
- *sizeCustomerQueue* số lượng khách hàng đang chờ
- Để tiện nên sử dụng chung *class customer* nên *customerQueue* không dùng *prev*

2 Task1 RED

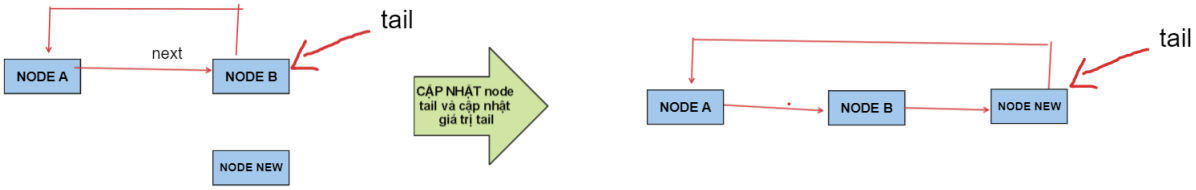
2.1 Hiện Thực

Mô tả: Hàm này có chứ năng là sắp xếp vị trí chỗ ngồi cho khách. Thực hiện theo các bước sau

1. Bước 1: Nhà hàng chỉ tiếp đón chú thuật sư (ENERGY dương) hoặc oán linh (ENERGY âm) và từ chối nhận khách hàng có ENERGY bằng 0. -> ENERGY bằng 0 đuổi khách về
2. Bước 2: Hàng chờ đã đạt đến số lượng tối đa thì nhà hàng sẽ ngưng nhận khách. -> Hàng chờ đầy đuổi khách về
3. Bước 3: Và bởi vì "thiên thượng thiên hạ, duy ngã độc tôn", nên nhà hàng sẽ không đón tiếp các vị khách đến sau và không cho phép vào hàng chờ nếu có tên giống với tên của các vị khách đang dùng bữa trong nhà hàng (hoặc đã có tên trong hàng chờ). Ví dụ một chú thuật sư tên ABC đang dùng bữa thì những chú thuật sư hoặc oán linh đến sau có tên ABC sẽ bị mời về. -> xét thử khách hàng trong hàng chờ và trong bàn có chung tên không, nếu chung thì đuổi khách về, dùng for duyệt từ 0 đến size nó và cập nhật khách hàng kế tiếp
4. Bước 4: Nếu số lượng khách ngồi vào bàn đã đạt đến MAXSIZE thì sẽ dừng việc nhận khách và đưa khách vào hàng chờ
   (a) Bước 4.1: xét xem trong hàng chờ có người nào không -> chèn trường hợp size=0 tương đương với chèn lần đầu danh sách liên kết vòng
   (b) Bước 4.2: Thêm khách hàng vào cuối hàng chờ thêm tại ví trí cuối tail
   (c) Chú ý có thể sử dụng hàng đợi bằng danh sách liên kết đơn (thêm tail vào), của anh là danh sách liên kết vòng chỉ có phần tư tail
5. Bước 5: Khách hàng đầu tiên vào quán -> Thêm khách hàng vào danh sách và cập nhật customerX và size, tương đương chèn lần đầu danh sách liên kết đôi vòng, **Phần này có thể lên trên bước 3**
6. Bước 6: Tuy nhiên, khi số lượng khách trong bàn lớn hơn hoặc bằng MAXSIZE/2. Nhân viên sẽ đổi chiến lược chọn chỗ ngồi cho khách. Vì biết những người có ENERGY gần bằng nhau thường không thích ngồi gần nhau. Do đó trước khi chọn vị trí, nhân viên sẽ tính giá trị chênh lệch lớn nhất giữa vị khách mới với tất cả vị khách trong nhà hàng thông qua việc lấy hiệu trị tuyệt đối của từng cặp ENERGY ứng với từng khách hàng (giả sử gọi là RES) để quyết định chỗ ngồi. -> Nếu vào trường hợp này cần tìm vị trí X mới, vì khi thay đổi X tại vị trí mớ ta có thể tận dụng được code bước 7, khi đó khách hàng có hiệu tuyệt đối ENEGY cực đại sẽ tương đương với khách hàng X, bước này dùng vòng for so sánh thôi
7. Bước 7: Khi vị khách tiếp theo đến, nhân viên sẽ bố trí chỗ ngồi cho khách tại vị trí liền kề bên phía thuận chiều kim đồng hồ nếu ENERGY của khách lớn hơn hoặc bằng với ENERGY của khách tại vị trí X. Ngược lại, khách sẽ ngồi tại vị trí liến kề bên phía ngược chiều kim đồng hồ của khách tại vị trí X -> Tạo Thông tin khách hàng mới sau đó cập nhật next prev của khách hàng mới khi thêm vào và cập nhật khác hàng trước khác hàng mới và sau khách hàng mới. Cập nhật CustomerX và size
   (a) thêm khách hàng mới vào phía trước khác hàng X, dùng x->next xử lí.
   (b) thêm khác hàng mới vào sau khách hàng X, dùng x->prev xử lí.



Hình 1: *Bước 7 hướng dẫn cách thêm node mới*



Hình 2: *Bước 4.2 hướng dẫn cách thêm node mới vào danh sách vòng*

2.2 Test Case

1. Test 1: xét bước 1, bước 3, bước 5. chú ý cái else cuối hàm Light đổi thành else if(number < 0)
2. Test 2: Chèn theo chiều kim đồng hồ của Bước 7 và print theo chiều kim đồng hồ
3. Test 3: Chèn theo ngược đồng hồ của Bước 7 và print theo chiều kim đồng hồ
4. Test 4: Chèn theo chiều kim đồng hồ của Bước 7 và print theo ngược chiều kim đồng hồ
5. Test 5: Chèn theo ngược đồng hồ của Bước 7 và print theo ngược chiều kim đồng hồ
6. Test 6: xét Bước 6 với MAXSIZE chẵn
7. Test 7: xét Bước 7 với MAXSIZE lẻ
8. Test 8: xét bước 4, bước 3, bước 2.
9. Test 9 -> 100: random.
10. phần này nếu chạy hết thì dùng lệnh dưới hướng dẫn tử [i,j] tương đương [1,100]

3 Task2 BLUE

3.1 Các Phần code bổ sung

```
//! class này dùng để quản lý thời gian của các khách hàng tới nhà hàng
//! data lưu khách hàng
class customerTime{
public:
        customer * data;
        customerTime* next;
public:
        customerTime(customer * data, customerTime* next = nullptr)
        :data(data),next(next) {}
};
private:
    customerTime * CustomerTimeHead; //! khách hàng đến lâu nhất
    customerTime * CustomerTimeTail; //! khách hàng đến mới nhất
```

- *class customerTime* dùng để lưu trữ xem khách nào tới trước khách nào tới sau
- *class customerTime* có thể xem như 1 node trong danh sách liên kết đơn
- *CustomerTimeHead* vị trí đầu tiên trong danh sách liên kết đơn, khách hàng đến lâu nhất
- *CustomerTimeTail* vị trí cuối trong danh sách liên kết đơn, khách hàng đến mới nhất

```
//^Thêm biến quản lý thời gian khách hàng nào đến trước bước 5
CustomerTimeTail = CustomerTimeHead = new customerTime (customerX);
return;
```

- phần này bổ xung tại cuối Bước 5
- Thêm khách hàng mới vào trong Danh sách thời gian, thêm khách hàng đầu tiên

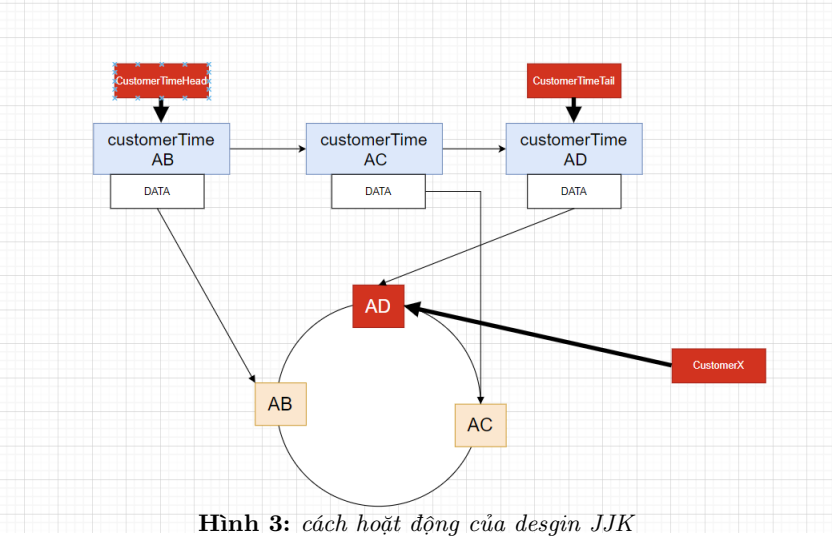
```
//^Thêm biến quản lý thời gian khách hàng nào đến trước cuối hàm
CustomerTimeTail->next = new customerTime (customerX,nullptr);
CustomerTimeTail = CustomerTimeTail->next;
return;
```

- phần này bổ xung tại cuối Hàm, cuối bước 7
- Thêm khách hàng mới vào trong Danh sách thời gian, khi số lượng khách trong bàn luôn có người

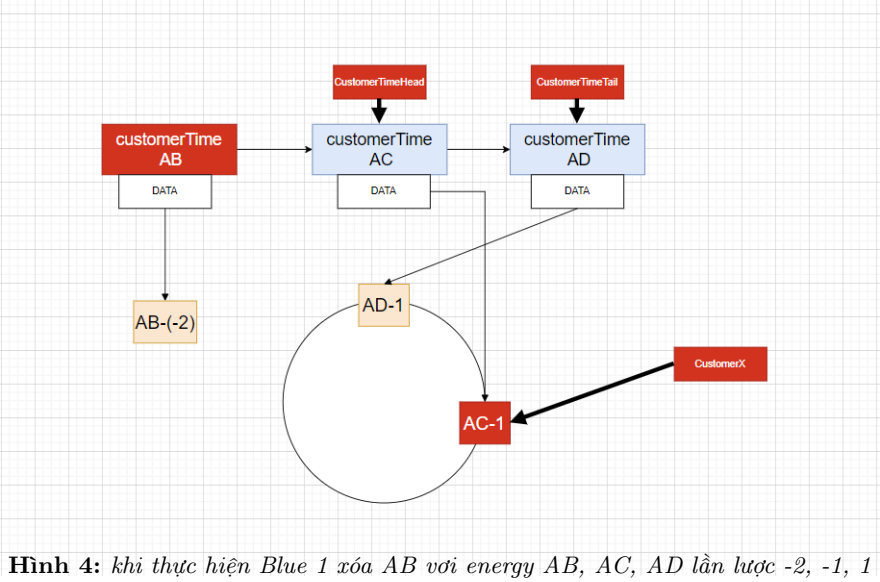
3.2 Giải thích kĩ về customerTime

Ý tưởng là sẽ lưu trữ các khách hàng vô lúc đầu đến vô lúc cuối cùng trong bàn ăn

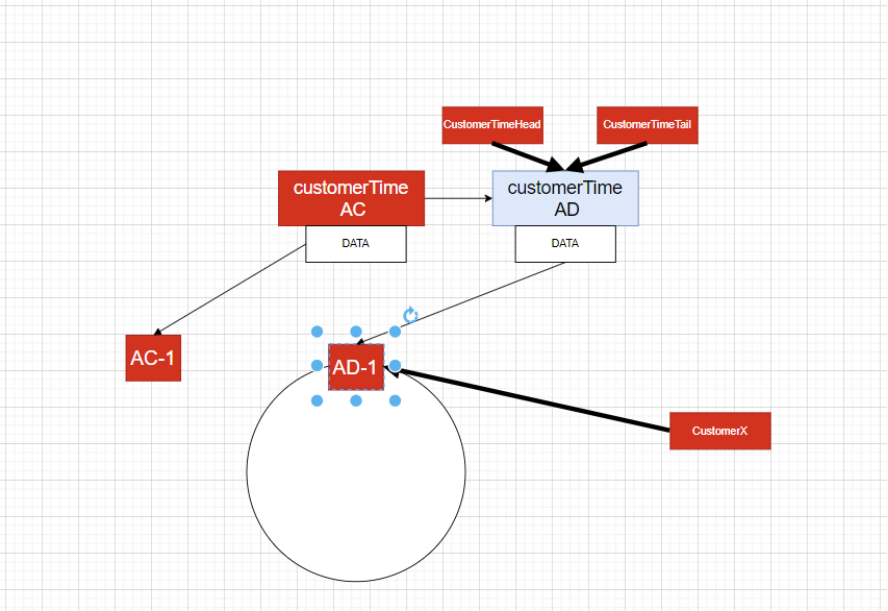
Mô tả: Hàm này có chức năng là đuổi khách hàng ở quá lâu. Thược Hiện theo các bước sau Image dưới mô tả khách hàng đầu tiên vào nhà hàng là AB khách hàng tới sau đó là AC và khách hàng cuối cung AD, khách hàng x đang làm AD, data của các customerTime của các khách hàng là class customerTime



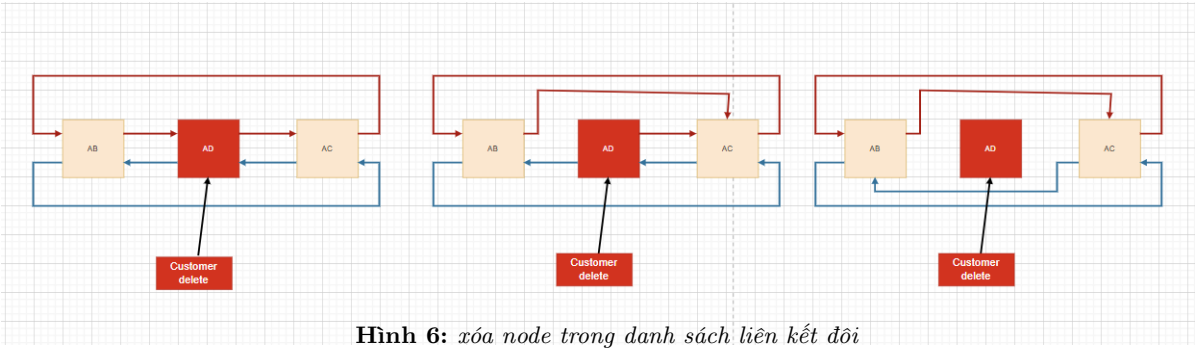
Hình 3: *cách hoạt động của desgin JJK*



Hình 4: *khi thực hiện Blue 1 xóa AB với energy AB, AC, AD lần lượt -2, -1, 1*



Hình 5: *khi thực hiện Blue 1 xóa AC với energy AB, AC, AD lần lượt -2, -1, 1*



Hình 6: *xóa node trong danh sách liên kết đôi*

3.3 Hiện Thực

1. Bước 1 Khi nhận được lệnh, nhân viên sẽ đuổi NUM vị khách theo thứ tự vào nhà hàng từ sớm nhất đến gần đây nhất. Ví dụ: khách đến nhà hàng theo thứ tự A → B → C → D thì sau khi thực thi lệnh BLUE 2 thì trong nhà hàng chỉ còn hai vị khách là C và D. Trường hợp nếu NUM lớn hơn hoặc bằng số lượng khách trong bàn ăn hoặc lớn hơn MAXSIZE thì xem như chủ nhà hàng quyết định đuổi hết khách trong bàn ăn. Dùng phòng for duyệt qua từng khách hàng bị đuổi.
   (a) Bước 1.1 đuổi khách hàng cuối cùng dùng để đuổi khách cuối cùng về gán các giá trị cần gán lại *nullptr* và thoát khỏi vòng for
   (b) Bước 1.2 Đuổi khách hàng bước này sẽ đuổi khách hàng trong bàn luôn lớn hơn 2 cập nhật lại customer trên bàn và trong danh sách liên kết CustomerTime. **-> Nếu mà khách có energy > 0 thì chọn khách hàng X là khách hàng phía trước của khách hàng chuẩn bị xóa, ngược lại (phần này trong fourm thầy có nói)** Hình 4 5 6
2. Bước 2 Sau khi dọn xong, nếu trong hàng chờ có khách, nhân viên sẽ tiến hành chọn chỗ cho khách, ngược lại không làm gì cả. Lưu ý: việc chọn chỗ ngồi mới cho khách trong hàng chờ chỉ được thực hiện sau khi đã thực hiện xong lệnh BLUE. Cơ chế chọn chỗ ngồi cho khách tương tự như lệnh RED và thứ tự của khách được xem xét ngồi vào bàn từ hàng chờ sẽ theo cơ chế First In First Out (FIFO). -> Tiến hành đưa khách hàng từ hàng chờ vào bàn ăn bằng vòng lặp for, thực hiện giảm số lượng khách hàng chờ và gọi hàm RED để thêm khách hàng mới vào
3. Một số hay sai 1: nhớ cập nhật size cho 2 bước
4. Một số hay sai 2: đối với bước 2 thì ta phải xóa trước khi thêm vào RED nếu không xóa trong hàng chờ thì khi thêm vô sẽ bị fail vì trùng tên với hàng chờ

3.4 Test Case

1. Test 101: xét bước 1.1 xem thử nếu xóa 1 khách hàng trong bàn có duy nhất 1 khách hàng có lỗi gì không.
2. Test 102: xét tại bước 1.2 xem thử xóa từng khách hàng trong bàn có lỗi gì không, thêm trước rồi xóa (không vừa thêm vừa xóa)
3. Test 103: xét tại bước 1.2 xét phần cập nhật lại khách hang X, và number > 1
4. Test 104: xóa khách hàng với number số lớn
5. Test 105 - 109: xét bước 2 khi thêm vào hàng chờ
6. 110 - 200: random
7. phần này nếu chạy hết thì dùng lệnh dưới hướng dẫn tử [i,j] tương đương [101,200]

4 Task3 DOMAIN_EXPANSION

4.1 Hiện Thực

1. Bước 1 Tổng ENERGY của tất cả chú thuật sư lớn hơn hoặc bằng tổng trị tuyệt đối ENERGY của tất cả chú linh có mặt tại nhà hàng (đang có mặt ở nhà hàng = tất cả khách hàng trong bàn ăn và trong hàng chờ) -> Dùng for duyệt qua các phần tử và tính tổng của 2 enenry của thú linh và chú thuật sư
2. Bước 2 thì nhân viên sẽ đuổi tất cả oán linh có đang mặt ở hàng chờ nhà hàng or tất cả thuật sư có đang có mặt ở hàng chờ nhà hàng
   (a) Bước 2.1 tách làm 2 danh sách oán linh và thuật sư ra thành 2 danh sách khác nhau trong hàng chờ với WizardQueueHead và WizardQueueTail là vị trí đầu cuối trong hàng chờ toàn là thuật sư, SpiritQueueHead và SpiritQueueTail là vị trí đầu cuối trong hàng chờ toàn là oán linh,
   (b) Bước 2.2 xóa danh sách oán linh trong hàng chờ or xóa danh sách Thuật sư trong hàng chờ -> mới đầu ta sẽ print danh sách xóa ngược lại bằng đệ quy, xóa từng phần tử trong danh sách của các khách hàng đang thua so với khách hàng bên kia (oán linh or thuật sư), cập nhật lại khách hàng thắng trong hàng chờ
3. Bước 3 thì nhân viên sẽ đuổi tất cả các oán linh đang có mặt ở trong bàn nhà hàng or tất cả thuật sư có đang có mặt ở trong bàn nhà hàng mặt ở nhà hàng.
   (a) Bước 2.1 tách làm 2 danh sách oán linh và thuật sư ra thành 2 danh sách khác nhau trong hàng chờ với WizardHead và WizardTail là vị trí đầu cuối trong bàn ăn toàn là thuật sư, SpiritHead và Spirit=Tail là vị trí đầu cuối trong bàn ăn toàn là oán linh,
   (b) Bước 2.2 xóa danh sách oán linh trong hàng chờ or xóa danh sách Thuật sư trong bàn ăn -> mới đầu ta sẽ print danh sách xóa ngược lại bằng đệ quy, xóa từng phần tử trong danh sách của các khách hàng đang thua so với khách hàng bên kia (oán linh or thuật sư), cập nhật lại khách hàng thắng trong bàn ăn **phần này giống phần task2 trong bước 1, khi xóa cần cập nhật data trong các node của customerTime, và cập nhật lại khách hàng X**
4. Bước 4 Sau đó nếu có vị trí trống, nhân viên sẽ tiếp tục bố trí khách hàng trong hàng chờ vào bàn ăn -> này giống bước 2 hàm xóa kéo xuống làm như hệ là được

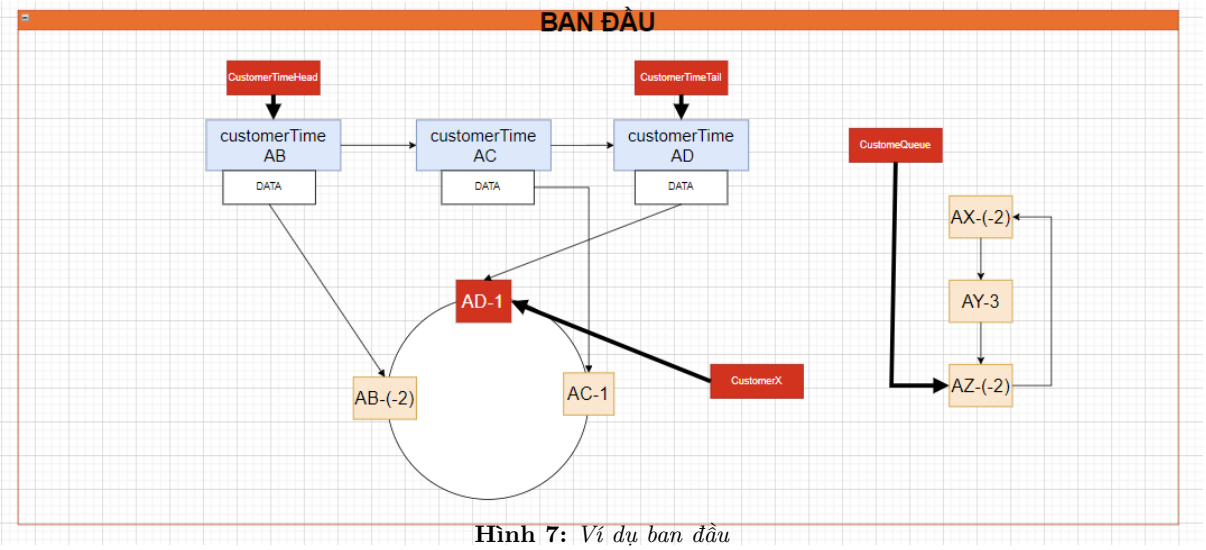
4.2 Test case

1. Test 201 : xét trường hợp xóa bình thường và TH khi 2 giá trị enery bằng nha
2. Test 202 : xét xóa hết toàn bộ khách trong ăn trong bàn, Hàng chờ đầy
3. Test 203 : ngược lại.
4. Test 204 - 205 : Hàng chờ bình thường không đầy
5. Test 206-207-208-209 : có hàm BLUE
6. Text 210-300 : random

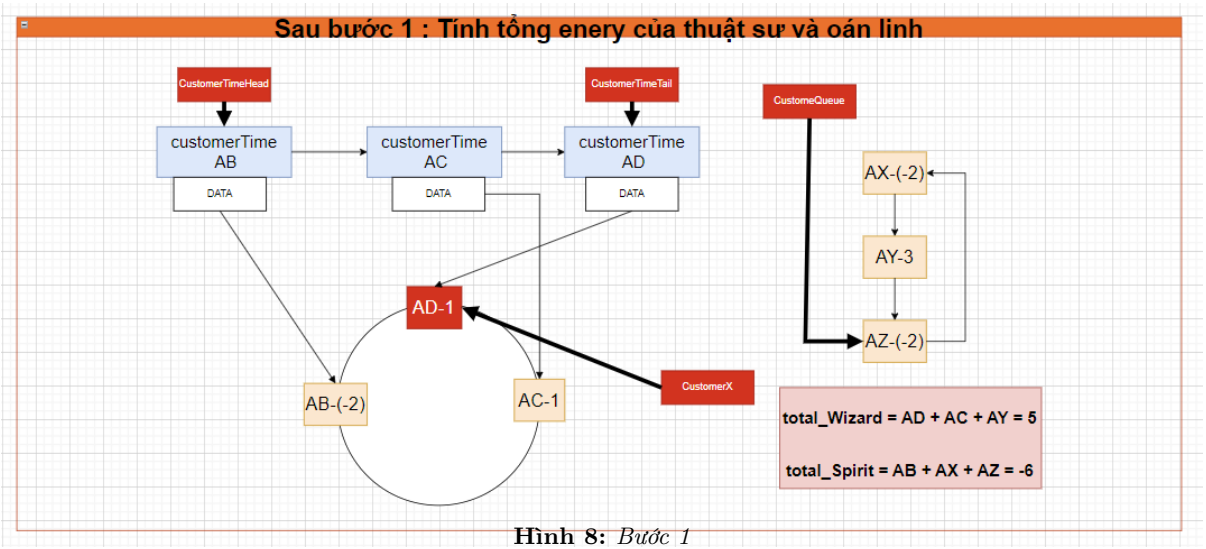
4.3 Ví dụ

Cho ban đầu có 3 khách hàng trong bàn theo thứ tự thêm vào (AB,enery = -2), (AC,enery = 1), (AD,enery = 1) và trong hàng đợi cũng 3 khách hàng theo thứ tự (AX,enery = -2), (AY,enery = 3), (AZ,enery = -2).

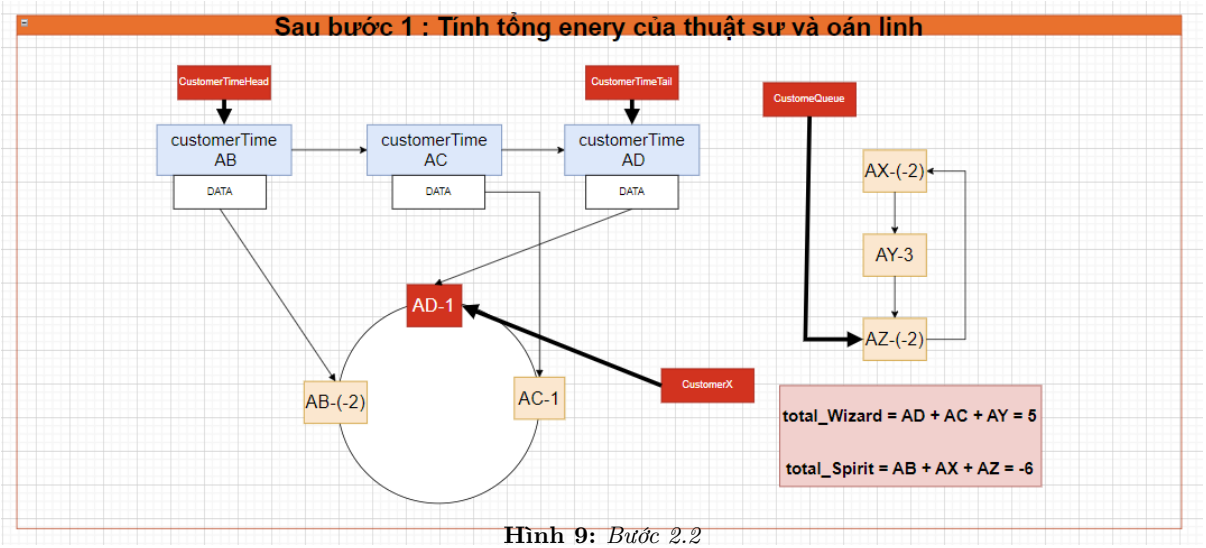
1. Bước 1: tính tổng enery $total\_Wizard$ = AD + AC + AY = 5, $total\_Spirit$ = AB + AX + AZ = -6
2. Bước 2: thu được hàng đợi $AX -> AZ$, xóa AY
3. Bước 3: thu được danh sách trong bàn AB
4. Bước 4 : thêm từ hàng chờ vào thu được $AB -> AX -> AZ$



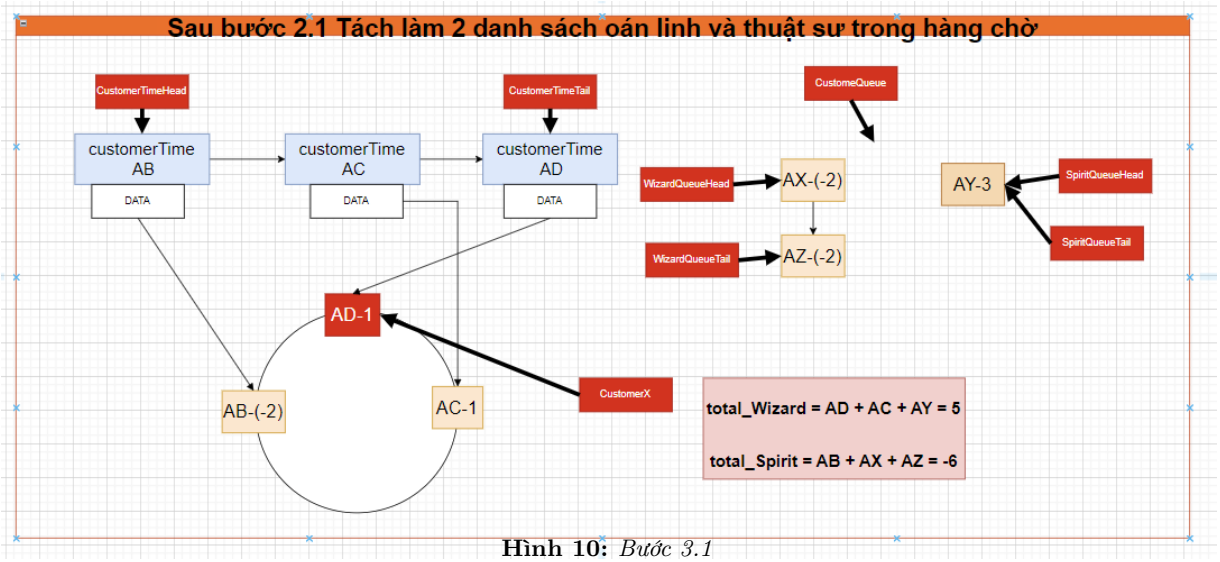
Hình 7: *Ví dụ ban đầu*



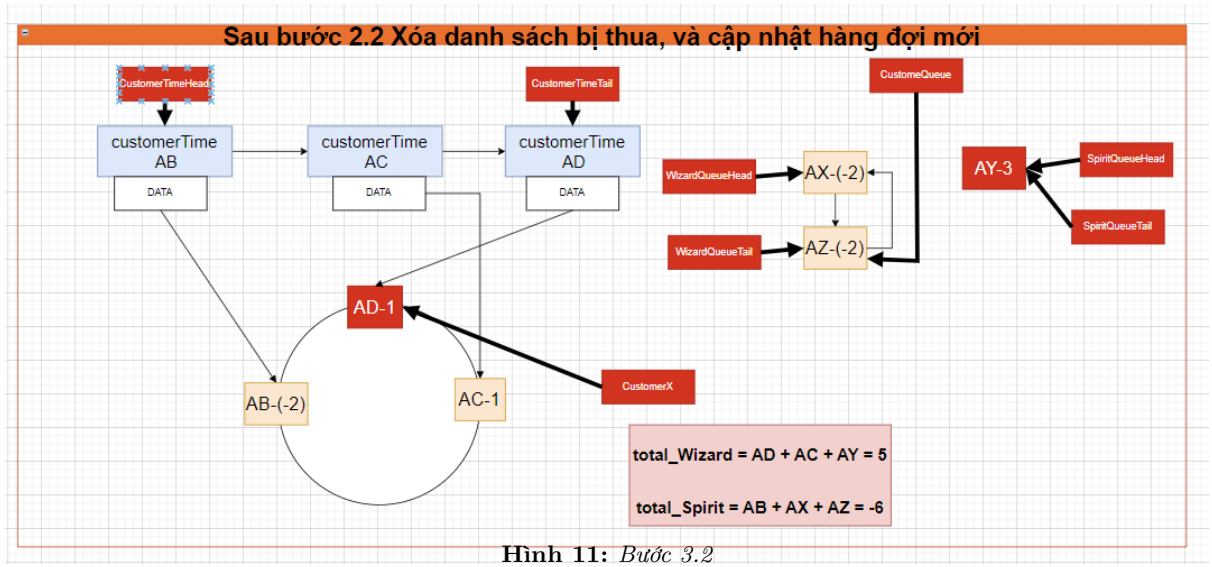
Hình 8: *Bước 1*



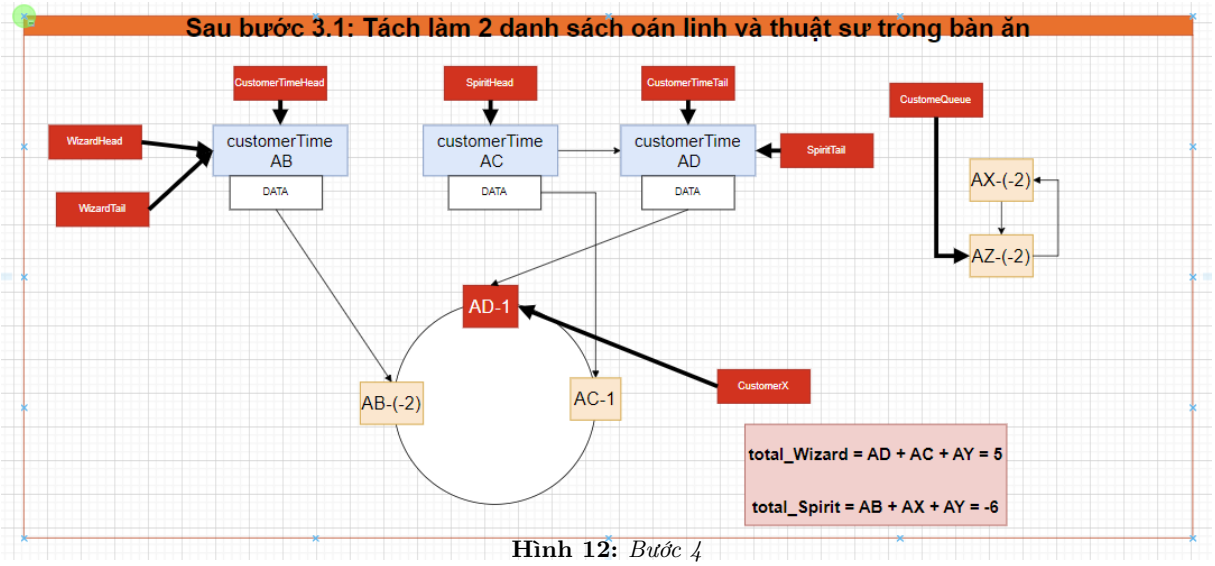
Hình 9: *Bước 2.2*



Hình 10: *Bước 3.1*



Hình 11: *Bước 3.2*



Hình 12: *Bước 4*

5 Hướng dẫn chạy test case

1. Yêu cầu cài $g++$, IDE Vscode (Vs có nhiều ràng buộc về code nên ít sài sài Vscode cho dễ), cài extensions vscode cài better comment
2. Tải từng task về tại đây **BTL**
3. Mở *new Termial* lên trong vscode
4. gõ lệnh g++ -o main main.cpp sau đó
   (a) Nếu muốn chạy từng test case gõ lênh .\main number với number là test case luôn chạy
   (b) Nếu muốn chạy trong 1 đoạn từ [i,j] gõ lênh .\main i j
5. Sẽ có video hương dẫn kĩ hơn