REPORT 24120184_WEEK6

1.Hàm vector<Company> readCompanyList(string file_name):

- Ở hàm này ta đọc từ file "MST.txt" và lấy dữ liệu 1250 công ty để lưu vào trong vector.
- Ý tưởng và cách thức thực hiện:

  -Ta duyệt từng dòng của của file txt, mỗi dòng lưu tương ứng dữ liệu cần thiết của mỗi công ty như: tên, lợi nhuận, địa chỉ.

  -Ta cần bỏ qua dòng đầu tiên, bắt đầu duyệt từ dòng thứ 2 lưu dữ liệu các công ty vào trong 1 biến string.

  -Duyệt từng phần tử của biến đó cho đến khi gặp kí tự '|' ,các kí tự trước đó chính là tên của công ty sau đó ta thêm vào vector.

  -Duyệt tương tự đến khi gặp kí tự '|', chuỗi thu được là dữ liệu lợi nhuận của công ty và ta thêm vào vector.

  -Sau cùng ta duyệt đến phần tử cuối cùng, chuỗi thu được là dữ liệu địa chỉ của công ty , sau đó ta thêm vào vector.

  -Trả về mảng vector sau khi thu được dữ liệu của các công ty.

2.Hàm long long hashString(string company_name):

- Hàm hash theo công thức:

$$hash(s) = \left(\sum_{i=0}^{n-1} (s[i] \times p^i) \right) \ mod \ m$$

  -Nếu tên có hơn 20 kí tự ta chỉ lấy 20 kí tự cuối, s[i] theo mã ASCII .

  -Cho p = 31 và m = 2000.

- Ở đây ta thấy:

  -Nếu p^{20} tức 31^{20} thì sẽ tràn số nên ta cần xử lý:

  +Ta có: (a + b) mod c bằng (a mod c + b mod c) mod c

  Vd: (5 + 2) mod 3 = 1 và (5 mod 3 + 2 mod 3) mod 3 = 4 mod 3 = 1.

  +Ta có: (a * b) mod c bằng ((a mod c)*b) mod c).

  Vd: (5 * 3) mod 4 = 3 và ((5 mod 4) * 3) mod 4 = 3 mod 4 = 3.

  -Ta xử lý phần mũ p^i thông qua hàm int modPow(int a, int b,int c):

  +Với p tương ứng là a, i tương ứng là b, m tương ứng là c.

  +Nếu i bằng 0 thì trả về 1.

+ i chạy từ 0 đến i: Cho kết quả ban đầu là 1, sau mỗi lần chạy cho kết quả nhân với p sau đó chia lấy dư cho m.

-Nếu tên có trên 20 kí tự ta chỉ xét 20 kí tự cuối, ngược lại ta xét toàn chuỗi:

+Duyệt từ kí tự đầu đến kí tự cuối: cho kết quả thu được cộng với (s[i] nhân cho kết quả thu được ở hàm modPow) mod m.

+Sau đó khi duyệt ta chia lấy dư cho m và trả về kết quả.

3.Hàm void insert(HashTable &hash_table, Company company):

- Viết hàm insert để thuận tiện hơn khi viết hàm createHashTable:
- Ta tìm vị trí chèn bằng hàm hashString thông qua tên của công ty đó:

+Nếu vị trí đó rỗng ta sẽ lưu dữ liệu của công ty vào vị trí tương ứng.

+Nếu không trống ta sẽ duyệt ở vị trí tiếp theo cho đến khi tìm được (Khi duyệt đến ô cuối thì ta quay lại vị trí đầu tiên là 0 để duyệt tiếp), nếu quay trở lại vị trí cũ tức là mảng băm đã đầy không thể chèn thêm được nữa.

4.Hàm HashTable createHashTable(vector<Company> list_company):

- Tạo 1 hashTable từ 1 danh sách công ty tương ứng, ta duyệt qua tên của từng công ty sau đó thêm vào mảng băm bằng hàm insert.

5.Hàm Company search(HashTable &hash_table, string company_name):

- Hàm tìm kiếm dữ liệu của 1 công ty trong mảng băm thông qua tên
- Ta cần tìm vị trí bằng hashString thông qua tên của công ty, tìm lần lượt từng vị trí cho đến khi tên nằm trong mảng băm giống với tên ta cần tìm, và trả về dữ liệu của công ty đó. (Khi duyệt đến ô cuối thì ta quay lại vị trí đầu tiên là 0 để duyệt tiếp).
- Nếu tên cần tìm không nằm trong mảng băm tức là ta tìm cho đến khi quay lại vị trí ban đầu (không tìm được) trả về “”.

Ảnh up code lên github: