大数据项目实验报告

一、项目概述

课程的大项目:位置相关服务:附近的出租车

位置相关服务:Location-Based Services (LBS) 利用定位技术获得移动终端(人)的位置信息，并通过通信网络 向移动终端(人)提供与位置相关的信息服务

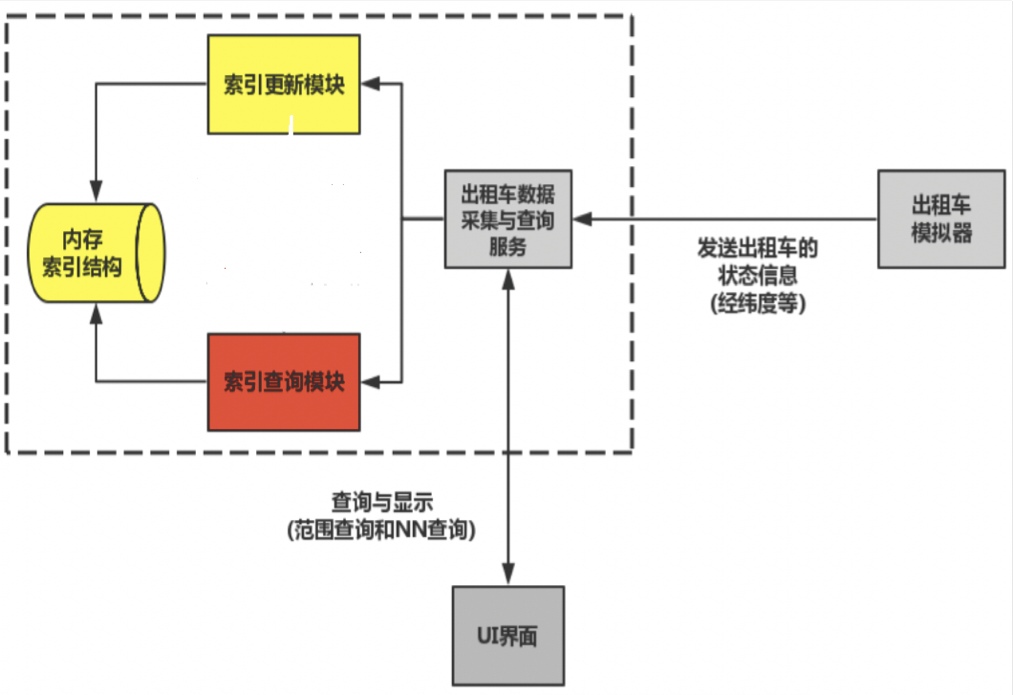
目标 ： 随时(Anytime)、随地(Anywhere)为所有的人(Anybody)和事(Anything)提供实时的“4A”服务

项目介绍： 以北京市出租车移动轨迹数据作为切入点，实现项目“位置相关服务:附近的 出租车”，目的是实现一个基于位置信息的查询服务，通过位置相关的持续查 询技术，实时回答用户:“我附近的出租车”，以此加深大家对相关领域理解 并提高实践能力。

项目主要内容：空间索引技术 利用空间索引有效地管理出租车的位置信息，并加快检索数据。 近邻查询技术 查询某一时刻处于某个地理区域的移动对象。 查询离某一点最近的移动对象

二、系统设计

系统环境：



模拟器起停

**启动**

cd lbs

./runKsimulator.sh

停止

cd lbs

lbs服务运行

./stop-simulator.sh • **lbs**

启动

• cd lbs

• ./run-server.sh

停止

• cd lbs

• ./stop-server.sh

三、功能设计分工

梅子豪：出租车位置更新，堆结构的建立。

陈润泽：建立hash表，建堆，网格与点的位置关系函数建立。

杨若云：网格位置索引，位图索引。

共同完成：大循环；公共模块

四、代码设计

lbs\_index.cpp

#include "server/grid/lbs\_index.h"

#include <stdio.h>

#include <stdlib.h>

#include "common/log.h"

#include "server/grid/lbs\_distance.h"

#define LBS\_LON\_MIN 116

#define LBS\_LON\_MAX 117

#define LBS\_LAT\_MIN 39

#define LBS\_LAT\_MAX 41

#define LBS\_ROW\_NUM 200

#define LBS\_COL\_NUM 100

#include "lbs\_grid.h"

#include "get\_cell\_id.h"

#include "calculate\_distance.h"

#include "lbs\_bitmap.h"

#include "SERVER\_GRID\_LBS\_NN\_HEAP.h"

static lbs\_grid\_t lbs\_grid;

// 初始化网格索引

int lbs\_grid\_index\_init() {

lbs\_grid\_init(&lbs\_grid, LBS\_LON\_MIN, LBS\_LON\_MAX, LBS\_LAT\_MIN, LBS\_LAT\_MAX, LBS\_ROW\_NUM, LBS\_COL\_NUM);

return 0;

}

// 更新接口[出租车位置更新]

int lbs\_grid\_index\_update(double lon,

double lat,

uint64\_t timestamp,

uint32\_t id) {

lbs\_grid\_update(&lbs\_grid, lon, lat, timestamp, id);

// TODO: by student

return 0;

}

// 范围查询接口[查询某一范围内的所有出租车信息]

int lbs\_grid\_index\_range\_query(double lon1,

double lon2,

double lat1,

double lat2,

lbs\_res\_node\_t\* out) {

int row1= lbs\_grid\_cell\_row(&lbs\_grid,lat1);

int col1= lbs\_grid\_cell\_col(&lbs\_grid,lon1);

int row2= lbs\_grid\_cell\_row(&lbs\_grid,lat2);

int col2= lbs\_grid\_cell\_col(&lbs\_grid,lon2);

for(int row=row1;row<=row2;row++)

{

for(int col=col1;col<=col2;col++)

{

int cell\_id = lbs\_grid\_cell\_id(&lbs\_grid,row,col);

lbs\_mov\_node\_t \*head = &(lbs\_grid.cell[cell\_id].dammy\_node);

lbs\_mov\_node\_t \*current = (lbs\_mov\_node\_t \*)head->queue.next;

while(current!=head){

if(current->lon>=lon1 && current->lon<=lon2 && current->lat>=lat1 && current->lat<=lat2){

lbs\_res\_node\_t\* node = (lbs\_res\_node\_t\*)malloc(sizeof(lbs\_res\_node\_t));

node->lon= current->lon;

node->lat= current->lat;

lbs\_queue\_insert\_head(&(out->queue), &(node->queue));

}

current = (lbs\_mov\_node\_t\*)current->queue.next;

}

}

}

// TODO: by student

return 0;

}

// NN查询接口[查询离lon,lat最近的出租车]

int lbs\_grid\_index\_nn\_query(double lon, double lat, lbs\_res\_node\_t\* out) {

// TODO: by student

//计算查询q所在的网格

int row=lbs\_grid\_cell\_row(&lbs\_grid,lat);

int col=lbs\_grid\_cell\_col(&lbs\_grid,lon);

int cell\_id=lbs\_grid\_cell\_id(&lbs\_grid,row,col);

//Mindist=0

int Mindist=lbs\_min\_distance(&lbs\_grid,cell\_id,lon,lat);

lbs\_nnheap\_t lbs\_nnheap;

lbs\_bitmap\_t lbs\_bitmap;

//创建小顶堆

lbs\_nnheap\_init(&lbs\_nnheap);

lbs\_bitmap\_init(&lbs\_bitmap, lbs\_grid.row\_num \* lbs\_grid.col\_num);

lbs\_bitmap\_setbit(&lbs\_bitmap, cell\_id);

printf("start insert heap\n");

lbs\_nnheap\_insert(&lbs\_nnheap, NULL, cell\_id, 1, Mindist);

printf("end insert heap\n");

while ((&lbs\_nnheap)->heap\_nodes[0].is\_grid != 0) {

int grid\_cell\_id = (&lbs\_nnheap)->heap\_nodes[0].cell\_id;

printf("start pop\n");

lbs\_nnheap\_pop(&lbs\_nnheap);

printf("end pop\n");

/// Step1: insert cars in cell

lbs\_cell\_t\* head =lbs\_grid\_cell(&lbs\_grid, grid\_cell\_id);

lbs\_mov\_node\_t \*cur = (lbs\_mov\_node\_t\*)head->dammy\_node.queue.next;

while(cur!= &(head->dammy\_node))

{ //q与空间点的距离

double dis=lbs\_distance(lon,lat, cur->lon, cur->lat) ;

//将空间点插入小顶堆

printf("start insert heap\n");

lbs\_nnheap\_insert(&lbs\_nnheap, cur, grid\_cell\_id, 0, dis);

printf("end insert heap\n");

cur= (lbs\_mov\_node\_t\*)cur->queue.next;

}

/\*求周围的cell\_id\*/

int a[8];

lbs\_get\_cell\_id(&lbs\_grid, grid\_cell\_id, a);

/\*计算到周围格子的距离并放入堆\*/

for(int i=0; i<8; i++){

double dis = lbs\_min\_distance(&lbs\_grid,a[i],lon,lat);

if(lbs\_bitmap\_isset(&lbs\_bitmap, a[i]) == 0) {

lbs\_nnheap\_insert(&lbs\_nnheap, NULL,a[i],1, dis);

lbs\_bitmap\_setbit(&lbs\_bitmap,a[i]);

}

}

printf("grid\_cell\_id=%d\n", grid\_cell\_id);

}

lbs\_res\_node\_t\* node = (lbs\_res\_node\_t\*)malloc(sizeof(lbs\_res\_node\_t));

node->lon = lbs\_nnheap.heap\_nodes[0].node->lon;

node->lat = lbs\_nnheap.heap\_nodes[0].node->lat;

lbs\_queue\_insert\_head(&(out->queue), &(node->queue));

return 0;

//return &lbs\_nnheap->heap\_nodes[0];

}

hash表：

#include "lbs\_hashtable.h"

#include <stdlib.h>

int lbs\_hashtable\_init(lbs\_hashtable\_t\* lbs\_hashtable) {

lbs\_hashtable->capacity = 20000;

lbs\_hashtable->size = 0;

lbs\_hashtable->hash\_nodes = (lbs\_hashnode\_t\*)malloc(sizeof(lbs\_hashnode\_t) \* lbs\_hashtable->capacity);

for (int i = 0; i < lbs\_hashtable->capacity; i++) {

lbs\_queue\_init(&(lbs\_hashtable->hash\_nodes[i].queue));

}

return 0;

}

int lbs\_hashtable\_destory(lbs\_hashtable\_t\* lbs\_hashtable) {

free (lbs\_hashtable->hash\_nodes);

lbs\_hashtable->hash\_nodes = NULL;

lbs\_hashtable->capacity = 0;

lbs\_hashtable->size = 0;

return 0;

}

int hash(uint32\_t id) {

return id%20000;

}

int lbs\_hashtable\_set(lbs\_hashtable\_t\* lbs\_hashtable, uint32\_t id, lbs\_mov\_node\_t\* lbs\_move\_node, int cell\_id) {

int index = hash(id);

lbs\_hashnode\_t\* node = (lbs\_hashnode\_t\*)malloc(sizeof(lbs\_hashnode\_t));

node->cell\_id = cell\_id;

node->mov\_node = lbs\_move\_node;

lbs\_queue\_insert\_head(&(lbs\_hashtable->hash\_nodes[index].queue) , &(node->queue));

return 0;

}

lbs\_hashnode\_t\* lbs\_hashtable\_get(lbs\_hashtable\_t\* lbs\_hash\_table, uint32\_t id) {

int index = hash(id);

lbs\_hashnode\_t \*head = (lbs\_hashnode\_t\*)&(lbs\_hash\_table->hash\_nodes[index].queue);

lbs\_hashnode\_t \*current = (lbs\_hashnode\_t\*)head->queue.next;

while(head != current) {

if (current->mov\_node->id == id) {

return current;

}

current = (lbs\_hashnode\_t\*)current->queue.next;

}

return NULL;

}

位置索引：

#include"server/grid/lbs\_hashtable.h"

#include "server/grid/lbs\_queue.h"

#include "server/grid/lbs\_grid.h"

#include <stdlib.h>

int lbs\_grid\_init(lbs\_grid\_t\* lbs\_grid, double lon1, double lon2, double lat1,double lat2, int row\_num, int col\_num)

{

lbs\_grid->row\_num=row\_num;

lbs\_grid->col\_num=col\_num ;

lbs\_grid->cell\_width=(lon2-lon1)/col\_num;

lbs\_grid->cell\_height=(lat2-lat1)/row\_num;

lbs\_grid->lon\_min=lon1;

lbs\_grid->lat\_min=lat1;

lbs\_hashtable\_init(&(lbs\_grid->hash\_table));

lbs\_grid->cell=(lbs\_cell\_t\*)malloc(row\_num\*col\_num\*sizeof(lbs\_cell\_t));

for(int i=0;i<row\_num\*col\_num;i++)

{

lbs\_queue\_init(&(lbs\_grid->cell[i].dammy\_node.queue));

}

return 0;

}

int lbs\_grid\_cell\_row(lbs\_grid\_t\* lbs\_grid,double lat) {

int cell\_row;

cell\_row=(lat-lbs\_grid->lat\_min)/lbs\_grid->cell\_height;

return cell\_row;

}

int lbs\_grid\_cell\_col(lbs\_grid\_t\* lbs\_grid,double lon)

{

int cell\_col;

cell\_col= (lon-lbs\_grid->lon\_min)/lbs\_grid->cell\_width;

return cell\_col;

}

int lbs\_grid\_cell\_id(lbs\_grid\_t\* lbs\_grid,int cell\_row,int cell\_col)

{ int cell\_id;

cell\_id= lbs\_grid->col\_num \* cell\_row + cell\_col;

return cell\_id;

}

void lbs\_grid\_cell\_row\_col(lbs\_grid\_t\* lbs\_grid,int cell\_id,int\*cell\_row,int\*cell\_col)

{

\*cell\_row=cell\_id/lbs\_grid->col\_num;

\*cell\_col=cell\_id%lbs\_grid->col\_num;

}

lbs\_cell\_t\* lbs\_grid\_cell(lbs\_grid\_t\* lbs\_grid, int cell\_id)

{

return lbs\_grid->cell+cell\_id;

}

位置更新：

#include"lbs\_grid.h"

#include <stdlib.h>

int lbs\_grid\_update(lbs\_grid\_t\* lbs\_grid,double lon,double lat,uint64\_t timestamp,uint32\_t id)

{

lbs\_hashnode\_t\* p;

lbs\_mov\_node\_t\* q;

//计算lon lat对应的col row

int row1 = lbs\_grid\_cell\_row(lbs\_grid, lat);

int col1 = lbs\_grid\_cell\_col(lbs\_grid, lon);

//算出lon lat对应的cell id

int cell\_id=lbs\_grid\_cell\_id(lbs\_grid, row1, col1);

//遍历hash表

p = lbs\_hashtable\_get(&lbs\_grid->hash\_table, id);

if (p)

{

//修改数据

p->mov\_node->lon=lon;

p->mov\_node->lat=lat;

p->mov\_node->timestamp=timestamp;

if (p->cell\_id != cell\_id)

{

lbs\_queue\_remove(&p->mov\_node->queue);

lbs\_queue\_insert\_head(&lbs\_grid->cell[cell\_id].dammy\_node.queue,

&p->mov\_node->queue);

p->cell\_id = cell\_id;

}

}

else//没找到就执行5

{

q=(lbs\_mov\_node\_t\*)malloc(sizeof(lbs\_mov\_node\_t));

q->lon=lon;

q->lat=lat;

q->id=id;

q->timestamp=timestamp;

lbs\_queue\_insert\_head(&lbs\_grid->cell[cell\_id].dammy\_node.queue,

&q->queue);

lbs\_hashtable\_set(&lbs\_grid->hash\_table, id, q, cell\_id);

}

return 0;

}

位图索引：

#include<stdlib.h>

#include "server/grid/lbs\_bitmap.h"

int arr[8]={128,64,32,16,8,4,2,1};

int lbs\_bitmap\_init(lbs\_bitmap\_t\* lbs\_bitmap, uint32\_t bits\_num)

{

lbs\_bitmap->bits\_num=bits\_num ;

if(bits\_num%8!=0) {

lbs\_bitmap->bits=(uint8\_t \*)calloc(1, (bits\_num/8+1)\*sizeof(uint8\_t));

} else {

lbs\_bitmap->bits=(uint8\_t \*)calloc(1, (bits\_num/8)\*sizeof(uint8\_t));

}

return 0;

}

int lbs\_bitmap\_destroy(lbs\_bitmap\_t\* lbs\_bitmap) {

free(lbs\_bitmap->bits);

}

int lbs\_bitmap\_setbit(lbs\_bitmap\_t\* lbs\_bitmap, uint32\_t pos)

{

int pos\_num= pos/8;

int pos\_id=pos%8;

lbs\_bitmap->bits[pos\_num]=lbs\_bitmap->bits[pos\_num]|arr[pos\_id];

return 0;

}

int lbs\_bitmap\_unsetbit(lbs\_bitmap\_t\* lbs\_bitmap, uint32\_t pos) { }

int lbs\_bitmap\_isset(lbs\_bitmap\_t\* lbs\_bitmap, uint32\_t pos)

{

int pos\_num= pos/8;

int pos\_id=pos%8;

if((lbs\_bitmap->bits[pos\_num]&arr[pos\_id])!=0)

{

return 1;

}else

{

return 0;

}

}

最近邻查询：

#include <stdio.h>

#include <stdlib.h>

#include"SERVER\_GRID\_LBS\_NN\_HEAP.h"

int lbs\_nnheap\_init(lbs\_nnheap\_t\* lbs\_nnheap) {

lbs\_nnheap->capacity = 100;

lbs\_nnheap->size = 0;

lbs\_nnheap->heap\_nodes = (lbs\_heapnode\_t\*)malloc(sizeof(lbs\_heapnode\_t) \* lbs\_nnheap->capacity);

return 0;

}

int lbs\_nnheap\_destroy(lbs\_nnheap\_t\* lbs\_nnheap) {

free(lbs\_nnheap->heap\_nodes);

lbs\_nnheap->heap\_nodes = NULL;

lbs\_nnheap->capacity = 0;

lbs\_nnheap->size = 0;

return 0;

}

int lbs\_nnheap\_insert(lbs\_nnheap\_t\* lbs\_nnheap,

lbs\_mov\_node\_t\* lbs\_mov\_node,

int cell\_id, uint8\_t is\_grid, double distance) {

if (lbs\_nnheap->size == lbs\_nnheap->capacity) {

lbs\_nnheap->capacity \*= 2;

lbs\_nnheap->heap\_nodes = (lbs\_heapnode\_t\*)realloc(lbs\_nnheap->heap\_nodes,

sizeof(lbs\_heapnode\_t) \* lbs\_nnheap->capacity);

}

lbs\_nnheap->heap\_nodes[lbs\_nnheap->size].is\_grid = is\_grid;

lbs\_nnheap->heap\_nodes[lbs\_nnheap->size].distance = distance;

lbs\_nnheap->heap\_nodes[lbs\_nnheap->size].cell\_id = cell\_id;

lbs\_nnheap->heap\_nodes[lbs\_nnheap->size].node = lbs\_mov\_node;

lbs\_nnheap->size++;

int i = lbs\_nnheap->size;

while(i > 1) {

if(lbs\_nnheap->heap\_nodes[i - 1].distance < lbs\_nnheap->heap\_nodes[i/2 - 1].distance){

lbs\_heapnode\_t a;

a = lbs\_nnheap->heap\_nodes[i - 1];

lbs\_nnheap->heap\_nodes[i - 1] = lbs\_nnheap->heap\_nodes[i/2 - 1];

lbs\_nnheap->heap\_nodes[i/2 - 1] = a;

i = i/2;

} else {

break;

}

}

return 0;

}

lbs\_heapnode\_t\* lbs\_nnheap\_top(lbs\_nnheap\_t\* lbs\_nnheap) {

return &lbs\_nnheap->heap\_nodes[0];

}

void lbs\_nnheap\_pop(lbs\_nnheap\_t\* lbs\_nnheap) {

int i =lbs\_nnheap->size;

lbs\_nnheap->heap\_nodes[0] = lbs\_nnheap->heap\_nodes[i - 1];

lbs\_nnheap->size = lbs\_nnheap->size - 1;

int a = 1;

while(2\*a <= i){

int min = 2\*a;

if(2\*a <= i - 1) {

if(lbs\_nnheap->heap\_nodes[2\*a - 1].distance > lbs\_nnheap->heap\_nodes[2\*a].distance) {

min = 2\*a + 1;

}

}

if(lbs\_nnheap->heap\_nodes[a - 1].distance > lbs\_nnheap->heap\_nodes[min-1].distance) {

lbs\_heapnode\_t b;

b = lbs\_nnheap->heap\_nodes[min - 1];

lbs\_nnheap->heap\_nodes[min - 1] = lbs\_nnheap->heap\_nodes[a - 1];

lbs\_nnheap->heap\_nodes[a - 1] = b;

a = min;

}

else{

return;

}

}

}

五、运行结果

基本符合项目要求，能够查询到某一点附近的出租车数量，比较成功的解决问题。

六、心得体会

首先，关于这次小组活动，前期我负责的是“位置更新”这一板块，并且在后两天进行了“建堆”，但是由于队友没有考虑到我们的任务分配问题，热心的将建堆任务完成，于是我们小组建堆的代码用的是队友的代码；我们三个人共同完成了大循环和其他的任务。我认为我们这个分工十分的合理，不仅给了我们充分的独立思考的时间，而且让我们在合作中解决了这个问题。

其次，这次的活动真的是让我获益匪浅，我们小组三个人都不是计算机专业的，所以在实习开始前，我们都恶补了很多的相关知识，来了以后也是每天晚上自己查阅资料进行学习，说到这里，我特别感谢两位老师，，他们的专业知识都特别的强，教学水平也很高，对于我们抛出的各种问题都能迎刃而解，所以这次的实习我们都进行的非常顺利，在老师以及同学们的帮助下，我们成功解决了这个问题，并且学习到了很多知识。

最后，我想说这次实习带给我的不仅是知识欠缺上的弥补，还有老师潜移默化传授的思维方式，这对我们今后的学习和研究都会起着重要的作用，能帮助我们在今后碰到其他困难时依然能够克服。还有，这次的实习经历也让我认识到了很多优秀的同学，让我在反省自身的同时，以他们为目标来努力学习。这次的学习之旅将成为我人生中一笔宝贵的财富。