

《数据结构》上机实验报告

第 8 次上机

学号： 201611130126

姓名： 袁宇昊

学院： 信息科学学院

专业： 计算机科学

教师： 郑新

日期： 2018.12.28

1. 实验要求

1.上机之前应做好充分准备，认真思考所需的上机题目，提高上机效率。

2.独立上机输入和调试自己所编的程序，切忌抄袭、拷贝他人程序。

3.上机结束后，整理出实验报告。书写报告时，重点放在实验的方法、思路以及总结反思上，以达到巩固课堂学习、提高动手能力的目的。

1. 实验内容

一．问题描述

对2006年度全国80多个城市的每天空气质量状况进行查询、排序等操作。

空气质量状况对象包括城市代码、城市名称、首要污染物、污染指数、污染物级别、空气状况、年、月、日

二、实验要求

1． 普通查询：输入城市名称和城市代码，分别查询该城市每天、每周、每月、每季度和全年的空气质量状况

例子：查询太原市2006年第8周的空气质量状况

2．统计查询：

（1）输入城市名称和城市代码，分别查询该城市每周、每月、每季度和全年的空气质量为优、良、轻微污染、轻度污染、重污染的天数

例子：查询石家庄市2006年第2季度空气质量为轻微污染的总天数

（2）根据时间查询空气质量状况：输入周编号、月编号、季度编号或年编号，以及空气质量为优、良、轻微污染、轻度污染、重污染的天数，查找相应的城市名称

例子：查询2006年5月，空气被轻度污染3天以上的城市有哪些？

3．排序查询

（1）输入周编号、月编号、季度编号或年编号，查询城市空气质量的排行榜

例子：查询2006年第6周，全国空气平均质量最好的前20个城市为哪些？

三、数据文件

Data.txt

1. 实验步骤（写出问题分析或者算法思路）
2. 因为问题涉及到的城市有77个，每个城市都有约一年的数据量，共有27263条数据，规模较大，所以用散列法来储存信息。

使用了inf类来表示每一条信息，它存有7个天气信息：城市号、城市名、主要污染物、空气质量得分、空气质量等级（罗马字）、空气质量等级（汉字）、还有日期（用date类表示）。

日期是用一个date类来表示，date类有年月日。这两个类都各自有自己的一些构造函数和其他功能函数。比如date里类的th()函数就是这个日期在当年的天序数（1月2日的th是2，2月1日的th是32等等）。

最后使用了D数组来储存整个天气信息。

1. 读入数据：从 Data.txt文件中使用从文件读入数据的办法读入数据。使用散列函数f来查找出信息对应于D数组种的位置，并进行了冲突处理。
2. 散列函数f的选取，注意到每一个天气信息由两点决定：1是城市，2是日期。散列函数的参数就可以确定下来是天气信息的城市编号和日期。又进一步发现两城市编号相差最小是100，所以最后选用(a.number\*8+a.time.th()\*2+a.time.day)%mod;作为基础的散列函数。a.number\*8就让任意两个城市间的散列值大于了800，所以有很好的分散度。mod的取值是第一个小于D数组大小（27262\*4）的素数，mod约等于（27262\*4），而天气信息有27262条，所以D散列数组的密度大约是1/4。在f函数中采用了线性勘察的办法来处理第一地址冲突的情况。最后有重叠的情况有4662次，远远小于27262条信息，并且说明查找一次信息的最差情况就是4662次访问D数组。在读入数据时，共访问了D数组127555次，所以查找一次信息的平均速度应该是127555/27262约等于4.5次。

这两个速度都是可以接受的。

1. 简单查询，最简单的想法就是配合散列函数，去查找信息。我的散列函数需要的只是城市编号和日期，在输入所求城市和日期后，经过处理，就可以用散列函数找到所求信息在D数组里的下标，再用下标访问就可以得到信息了。
2. 统计查询，在简单查询的基础上利用统计数组进行统计就可以了。

然后使用交互的方式来查询不同类型的信息。

1. 排序查询，在简单查询的基础上利用统计的办法得到每个城市的分数后再调用sort函数排序就可以了。
2. 程序清单（源程序代码等）

main.cpp

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
3. **const** **int**  monthday[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
4. **const** **int** mday[]={0,0,31,59,90,120,151,181,212,243,273,304,334,365};
6. **struct** item
7. {
8. string name;
9. **double** score;
10. **bool** operator < (item & a)
11. {
12. **return** score<=a.score;
13. }
14. };
15. item p[10000];
16. **class** date
17. {
18. **public**:
19. **int** year,month,day,week;
20. date(**int** y,**int** m,**int** d):
21. year(y),month(m),day(d){
22. week=(mday[month]+day-1)%7;
23. }
24. date(**int** th=1)
25. {
26. year=2006;
27. **for**(month=1;month<=12;month++)
28. {
29. **if**(th>mday[month+1])
30. **continue**;
31. **else**
32. **break**;
33. }
34. day=th-mday[month];
35. //cout<<month<<"月"<<day<<endl;
36. week=(mday[month]+day-1)%7;
37. }
38. **int** th(){
39. **return** mday[month]+day;
40. }
41. **bool** operator < (date a){
42. **return** th()<a.th();
43. }
44. **bool** operator == (date a){
45. **if**(year==a.year && month == a.month && day==a.day)
46. **return** **true**;
47. **return** **false**;
48. }
49. **friend** ostream& operator << (ostream& os,**const** date &a);
50. **friend** istream& operator >> (istream& is,date &a);
51. };
52. ostream& operator <<(ostream& os,**const** date &a)
53. {
54. os<<a.year<<"年"<<a.month<<"月"<<a.day<<"日";
55. **return** os;
56. }
57. istream& operator >> (istream& is,date &a)
58. {
59. is>>a.year>>a.month>>a.day;
60. **return** is;
61. }
62. **class** inf
63. {
64. **public**:
65. **int** number;     //320600
66. string name;    //南通
67. string main\_pollution;  //--
68. **int** score;      //43
69. string level2;  //I
70. string level;   //优
71. date time;      //2006 12 31
72. inf(**int** n=1,date a=date(0)){
73. number=n;
74. time=a;
75. }
76. **friend** ostream& operator << (ostream &os,**const** inf &a);
77. **friend** istream& operator >> (istream &os,inf &a);
78. **bool** operator == (inf a)
79. {
80. **return** number==a.number && a.time==time;
81. }
82. };
83. istream& operator >> (istream& is,inf &a)
84. {
85. is>>a.number>>a.name>>a.main\_pollution>>a.score>>a.level2>>a.level>>a.time;
86. **return** is;
87. }
88. ostream& operator <<(ostream& os,**const** inf &a)
89. {
90. **if**(a.time.day==0)
91. {
92. cout<<"没有数据"<<endl;
93. **return** os;
94. }
95. os<<a.time<<a.name<<"的空气质量为"<<a.level<<endl;
96. //os<<a.year<<"年"<<a.month<<"月"<<a.day<<"日"<<endl;
97. **return** os;
98. }
99. **int** find\_su(**int** N)
100. {
101. **int** flag=1;
102. **for**(**int** i=N;i>=2;i--)
103. {
104. flag=1;
105. **for**(**int** j=2;j<= sqrt(i);j++)
106. {
107. **if**(i%j==0)
108. {
109. flag=0;
110. **break**;
111. }
112. }
113. **if**(flag)
114. {
115. **return** i;
116. }
117. }
118. **return** -1;
119. }
120. **const** **int** maxsize=27262\*4;
121. **const** **int** mod=find\_su(maxsize);
122. **int** chongfu=0;
123. **int** used[maxsize];
124. inf D[maxsize];
125. **int** f(inf a)   //散列函数
126. {
127. **int** flag=0;
128. //  cout<<" "<<a.name<<a.time<<endl;
129. **int** zhongzi=(a.number\*8+a.time.th()\*2+a.time.day)%mod;
130. **while**(used[zhongzi])
131. {
132. **if**(D[zhongzi]==a)
133. **return** zhongzi;
134. //if(a.name=="石家庄")
135. //cout<<"重复"<<D[zhongzi].name<<D[zhongzi].time<<" "<<a.name<<a.time<<endl;
136. chongfu++;
137. zhongzi++;
138. zhongzi%=maxsize;
139. }
140. //chongfu+=flag;
141. **return** zhongzi;
142. }
143. map<string , **int**> name2number;
144. map<**int** , string> number2name;
145. map<string , **int**> level2number;
146. **int** th[5000];
147. **int** use[1000000]={0};
148. **int** citycnt=0;
149. string levelname[]={"优","良","轻微污染","轻度污染","重污染"};
150. **int** main ()
151. {
153. **for**(**int** i=0;i<5;i++)
154. {
155. level2number[levelname[i]]=i;
156. }
157. //cout<<mod<<endl;
158. **for**(**int** i=0;i<maxsize;i++)
159. use[i]=0;
160. inf a;
161. ifstream is("data.txt");
162. **int** mycnt=0;
163. **while**(is>>a)
164. {
165. mycnt++;
166. **if**(used[a.number]==0)
167. {
168. used[a.number]=1;
169. th[citycnt++]=a.number;
170. }
171. name2number[a.name]=a.number;
172. number2name[a.number]=a.name;
173. **int** pos=f(a);
174. D[pos]=a;
175. used[pos]=1;
176. }
178. cout<<"共"<<mycnt<<"条数据,包括了"<<citycnt<<"个城市"<<endl;
179. cout<<mod<<endl;
180. string name;
181. **int** Year,Month,Day,number,Week;
182. **char** flag='Y';
183. cout<<"普通查询:"<<endl;
184. **while**(flag=='Y')    // 普通查询
185. {
186. cout<<"输入查询的城市名称";
187. cin>>name;
188. number=name2number[name];
189. //cout<<number<<endl;
190. cout<<"输入查询的天序数";
191. **int** day;
192. cin>>day;
193. date t(day);
194. cout<<D[f(inf(number,t))];
196. cout<<"输入查询的周数";
197. cin>>Week;
198. **for**(**int** i=Week\*7-6;i<=Week\*7;i++)
199. {
200. **if**(i<=0 || i>=366)
201. **continue**;
202. date tmp(i);
203. //cout<<tmp<<endl;
204. //cout<<inf(number,tmp)<<endl;;
205. //cout<<"f"<<f(inf(number,tmp))<<endl;;
206. //cout<<inf(number,tmp)<<endl;
207. cout<<D[f(inf(number,tmp))];
208. }
210. cout<<"输入查询的月数";
211. **int** month;
212. cin>>month;
213. **for**(**int** i=mday[month]+1;i<=mday[month+1];i++)
214. {
215. **if**(i<=0 || i>=366)
216. **continue**;
217. date tmp(i);
218. cout<<D[f(inf(number,tmp))];
219. }
221. cout<<"输入查询的季度 （1 = 第1季度 = 1~3月）";
222. **int** season;
223. cin>>season;
224. **for**(**int** i=mday[season\*3-2]+1;i<=mday[season\*3+1];i++)
225. {
226. **if**(i<=0 || i>=366)
227. **continue**;
228. date tmp(i);
229. cout<<D[f(inf(number,tmp))];
230. }
231. cout<<"还要继续普通查询吗？（Y/N） N的话进入统计查询"<<endl;
232. cin>>flag;
233. }
235. flag='Y';
236. cout<<"统计查询（1）"<<endl;
237. **while**(flag=='Y')    // 统计查询1
238. {
239. cout<<"输入查询的城市名称";
240. cin>>name;
241. number=name2number[name];
242. //cout<<number<<endl;
243. **int** cnt[10]={0};
245. cout<<"输入查询的周数";
246. cin>>Week;
247. **for**(**int** i=Week\*7-6;i<=Week\*7;i++)
248. {
249. **if**(i<=0 || i>=366)
250. **continue**;
251. date tmp(i);
252. **if**(D[f(inf(number,tmp))].time.day==0)
253. {
254. cout<<tmp<<"没有数据"<<endl;
255. }
256. **else**
257. {
258. cnt[level2number[D[f(inf(number,tmp))].level]]++;
259. }
260. }
261. **for**(**int** i=0;i<5;i++)
262. {
263. cout<<levelname[i]<<" :\t"<<cnt[i]<<endl;
264. cnt[i]=0;
265. }
267. cout<<"输入查询的月数";
268. **int** month;
269. cin>>month;
270. **for**(**int** i=mday[month]+1;i<=mday[month+1];i++)
271. {
272. **if**(i<=0 || i>=366)
273. **continue**;
274. date tmp(i);
275. **if**(D[f(inf(number,tmp))].time.day==0)
276. {
277. cout<<tmp<<"没有数据"<<endl;
278. }
279. **else**
280. {
281. cnt[level2number[D[f(inf(number,tmp))].level]]++;
282. }
283. }
284. **for**(**int** i=0;i<5;i++)
285. {
286. cout<<levelname[i]<<" :\t"<<cnt[i]<<endl;
287. cnt[i]=0;
288. }
290. cout<<"输入查询的季度 （1 = 第1季度 = 1~3月）";
291. **int** season;
292. cin>>season;
293. **for**(**int** i=mday[season\*3-2]+1;i<=mday[season\*3+1];i++)
294. {
295. **if**(i<=0 || i>=366)
296. **continue**;
297. date tmp(i);
298. **if**(D[f(inf(number,tmp))].time.day==0)
299. {
300. cout<<tmp<<"没有数据"<<endl;
301. }
302. **else**
303. {
304. cnt[level2number[D[f(inf(number,tmp))].level]]++;
305. }
306. }
307. **for**(**int** i=0;i<5;i++)
308. {
309. cout<<levelname[i]<<" :\t"<<cnt[i]<<endl;
310. cnt[i]=0;
311. }
313. cout<<"全年查询"<<endl;
314. **int** cnt2=0;
315. **for**(**int** i=1;i<=365;i++)
316. {
317. date tmp(i);
318. **if**(D[f(inf(number,tmp))].time.day==0)
319. {
320. cnt2++;
321. //cout<<tmp<<"没有数据"<<endl;
322. }
323. **else**
324. {
325. cnt[level2number[D[f(inf(number,tmp))].level]]++;
326. }
327. }
328. **for**(**int** i=0;i<5;i++)
329. {
330. cout<<levelname[i]<<" :\t"<<cnt[i]<<endl;
331. cnt[i]=0;
332. }
333. cout<<"有"<<cnt2<<"天没有数据"<<endl;
335. cout<<"还要继续统计查询（1）吗？（Y/N） N的话进入统计查询（2）"<<endl;
336. cin>>flag;
337. }
339. cout<<"统计查询（2）"<<endl;
340. flag='Y';
341. string level;
342. **int** levelth;
343. **int** type,x;
344. **while**(flag=='Y')
345. {
346. **int** cnt[10]={0};
347. cout<<"输入查询的周数";
348. cin>>Week;
349. cout<<"输入查询的项目";
351. cin>>level;
352. levelth=level2number[level];
353. cout<<"输入查询的条件 type x （type 1: >=x 2: <=x 3: ==x）";
354. cin>>type>>x;
355. **for**(**int** k=0;k<citycnt;k++)
356. {
357. number=th[k];
358. **for**(**int** i=Week\*7-6;i<=Week\*7;i++)
359. {
360. **if**(i<=0 || i>=366)
361. **continue**;
362. date tmp(i);
363. **if**(D[f(inf(number,tmp))].time.day==0)
364. {
365. //cout<<tmp<<"没有数据"<<endl;
366. }
367. **else**
368. {
369. cnt[level2number[D[f(inf(number,tmp))].level]]++;
370. }
371. }
372. **switch**(type)
373. {
374. **case** 1: **if**(cnt[levelth]>=x)
375. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
376. **break**;
377. **case** 2: **if**(cnt[levelth]<=x)
378. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
379. **break**;
380. **case** 3: **if**(cnt[levelth]==x)
381. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
382. **break**;
383. }
384. **for**(**int** i=0;i<5;i++)
385. {
386. cnt[i]=0;
387. }
388. //cout<<endl;
389. }

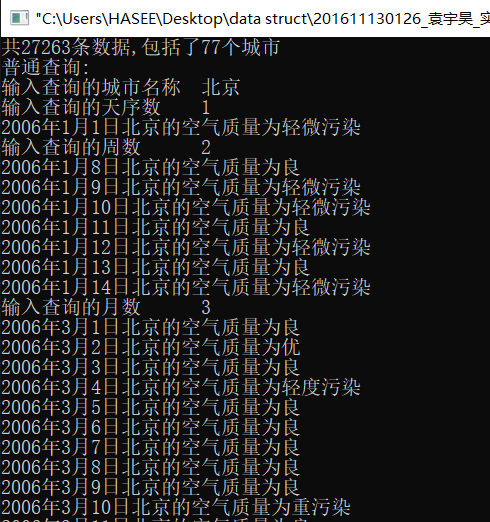
392. cout<<"\n输入查询的月数";
393. **int** month;
394. cin>>month;
395. cout<<"输入查询的项目";
396. //string level;
397. cin>>level;
398. levelth=level2number[level];
399. cout<<"输入查询的条件 type x （type 1: >=x 2: <=x 3: ==x）";
400. cin>>type>>x;
401. **for**(**int** k=0;k<citycnt;k++)
402. {
403. number=th[k];
404. **for**(**int** i=mday[month]+1;i<=mday[month+1];i++)
405. {
406. **if**(i<=0 || i>=366)
407. **continue**;
408. date tmp(i);
409. **if**(D[f(inf(number,tmp))].time.day==0)
410. {
411. //cout<<tmp<<"没有数据"<<endl;
412. }
413. **else**
414. {
415. cnt[level2number[D[f(inf(number,tmp))].level]]++;
416. }
417. }
418. **switch**(type)
419. {
420. **case** 1: **if**(cnt[levelth]>=x)
421. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
422. **break**;
423. **case** 2: **if**(cnt[levelth]<=x)
424. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
425. **break**;
426. **case** 3: **if**(cnt[levelth]==x)
427. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
428. **break**;
429. }
430. **for**(**int** i=0;i<5;i++)
431. {
432. cnt[i]=0;
433. }
434. //cout<<endl;
435. }
437. cout<<"\n输入查询的季度 （1 = 第1季度 = 1~3月）";
438. **int** season;
439. cin>>season;
440. cout<<"输入查询的项目";
441. //string level;
442. cin>>level;
443. levelth=level2number[level];
444. cout<<"输入查询的条件 type x （type 1: >=x 2: <=x 3: ==x）";
445. cin>>type>>x;
446. **for**(**int** k=0;k<citycnt;k++)
447. {
448. number=th[k];
449. **for**(**int** i=mday[season\*3-2]+1;i<=mday[season\*3+1];i++)
450. {
451. **if**(i<=0 || i>=366)
452. **continue**;
453. date tmp(i);
454. **if**(D[f(inf(number,tmp))].time.day==0)
455. {
456. //cout<<tmp<<"没有数据"<<endl;
457. }
458. **else**
459. {
460. cnt[level2number[D[f(inf(number,tmp))].level]]++;
461. }
462. }
463. **switch**(type)
464. {
465. **case** 1: **if**(cnt[levelth]>=x)
466. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
467. **break**;
468. **case** 2: **if**(cnt[levelth]<=x)
469. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
470. **break**;
471. **case** 3: **if**(cnt[levelth]==x)
472. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
473. **break**;
474. }
475. **for**(**int** i=0;i<5;i++)
476. {
477. cnt[i]=0;
478. }
479. //cout<<endl;
480. }
482. cout<<"\n全年查询"<<endl;
483. cout<<"输入查询的项目";
484. //string level;
485. cin>>level;
486. levelth=level2number[level];
487. cout<<"输入查询的条件 type x （type 1: >=x 2: <=x 3: ==x）";
488. cin>>type>>x;
489. **for**(**int** k=0;k<citycnt;k++)
490. {
491. number=th[k];
492. **for**(**int** i=1;i<=365;i++)
493. {
494. **if**(i<=0 || i>=366)
495. **continue**;
496. date tmp(i);
497. **if**(D[f(inf(number,tmp))].time.day==0)
498. {
499. //cout<<tmp<<"没有数据"<<endl;
500. }
501. **else**
502. {
503. cnt[level2number[D[f(inf(number,tmp))].level]]++;
504. }
505. }
506. **switch**(type)
507. {
508. **case** 1: **if**(cnt[levelth]>=x)
509. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
510. **break**;
511. **case** 2: **if**(cnt[levelth]<=x)
512. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
513. **break**;
514. **case** 3: **if**(cnt[levelth]==x)
515. cout<<number2name[number]<<'('<<cnt[levelth]<<')'<<"\t";
516. **break**;
517. }
518. **for**(**int** i=0;i<5;i++)
519. {
520. cnt[i]=0;
521. }
522. //cout<<endl;
523. }
525. cout<<"还要继续统计查询（2）吗？（Y/N） N的话进入排序查询"<<endl;
526. cin>>flag;
527. }
529. cout<<"\n排序查询\n";
530. flag='Y';
531. **while**(flag=='Y')
532. {
534. cout<<"输入查询的周数";
535. cin>>Week;
536. cout<<"输入查询前几位";
537. cin>>x;
538. **for**(**int** k=0;k<citycnt;k++)
539. {
540. number=th[k];
541. **int** sum=0;
542. **int** ok=0;
543. **for**(**int** i=Week\*7-6;i<=Week\*7;i++)
544. {
545. **if**(i<=0 || i>=366)
546. **continue**;
547. date tmp(i);
548. **if**(D[f(inf(number,tmp))].time.day==0)
549. {
550. //cout<<tmp<<"没有数据"<<endl;
551. }
552. **else**
553. {
554. sum+=D[f(inf(number,tmp))].score;
555. ok++;
556. }
557. }
558. **if**(ok>0)
559. {
560. p[k].name=number2name[number];
561. p[k].score=1.0\*sum/ok;
562. }
563. **else**
564. {
565. p[k].score=0;
566. p[k].name=number2name[number];
567. }
568. //cout<<endl;
569. }
570. sort(p,p+citycnt);
571. **for**(**int** i=0;i<x;i++)
572. {
573. cout<<p[i].name<<'('<<p[i].score<<')'<<"\t";
574. }
575. cout<<"\n";
577. cout<<"输入查询的月数";
578. **int** month;
579. cin>>month;
580. cout<<"输入查询前几位";
581. cin>>x;
582. **for**(**int** k=0;k<citycnt;k++)
583. {
584. number=th[k];
585. **int** sum=0;
586. **int** ok=0;
587. **for**(**int** i=mday[month]+1;i<=mday[month+1];i++)
588. {
589. **if**(i<=0 || i>=366)
590. **continue**;
591. date tmp(i);
592. **if**(D[f(inf(number,tmp))].time.day==0)
593. {
594. //cout<<tmp<<"没有数据"<<endl;
595. }
596. **else**
597. {
598. sum+=D[f(inf(number,tmp))].score;
599. ok++;
600. }
601. }
602. **if**(ok>0)
603. {
604. p[k].name=number2name[number];
605. p[k].score=1.0\*sum/ok;
606. }
607. **else**
608. {
609. p[k].score=0;
610. p[k].name=number2name[number];
611. }
612. //cout<<endl;
613. }
614. sort(p,p+citycnt);
615. **for**(**int** i=0;i<x;i++)
616. {
617. cout<<p[i].name<<'('<<p[i].score<<')'<<"\t";
618. }
619. cout<<"\n";
621. cout<<"输入查询的季度";
622. **int** season;
623. cin>>season;
624. cout<<"输入查询前几位";
625. cin>>x;
626. **for**(**int** k=0;k<citycnt;k++)
627. {
628. number=th[k];
629. **int** sum=0;
630. **int** ok=0;
631. **for**(**int** i=mday[season\*3-2]+1;i<=mday[season\*3+1];i++)
632. {
633. **if**(i<=0 || i>=366)
634. **continue**;
635. date tmp(i);
636. **if**(D[f(inf(number,tmp))].time.day==0)
637. {
638. //cout<<tmp<<"没有数据"<<endl;
639. }
640. **else**
641. {
642. sum+=D[f(inf(number,tmp))].score;
643. ok++;
644. }
645. }
646. **if**(ok>0)
647. {
648. p[k].name=number2name[number];
649. p[k].score=1.0\*sum/ok;
650. }
651. **else**
652. {
653. p[k].score=0;
654. p[k].name=number2name[number];
655. }
656. //cout<<endl;
657. }
658. sort(p,p+citycnt);
659. **for**(**int** i=0;i<x;i++)
660. {
661. cout<<p[i].name<<'('<<p[i].score<<')'<<"\t";
662. }
663. cout<<"\n";
665. cout<<"全年查询\n";
666. cout<<"输入查询前几位";
667. cin>>x;
668. **for**(**int** k=0;k<citycnt;k++)
669. {
670. number=th[k];
671. **int** sum=0;
672. **int** ok=0;
673. **for**(**int** i=1;i<=365;i++)
674. {
675. **if**(i<=0 || i>=366)
676. **continue**;
677. date tmp(i);
678. **if**(D[f(inf(number,tmp))].time.day==0)
679. {
680. //cout<<tmp<<"没有数据"<<endl;
681. }
682. **else**
683. {
684. sum+=D[f(inf(number,tmp))].score;
685. ok++;
686. }
687. }
688. **if**(ok>0)
689. {
690. p[k].name=number2name[number];
691. p[k].score=1.0\*sum/ok;
692. }
693. **else**
694. {
695. p[k].score=0;
696. p[k].name=number2name[number];
697. }
698. //cout<<endl;
699. }
700. sort(p,p+citycnt);
701. **for**(**int** i=0;i<x;i++)
702. {
703. cout<<p[i].name<<'('<<p[i].score<<')'<<"\t";
704. }
705. cout<<"\n";
707. cout<<"还要继续排序查询吗？（Y/N） N的话推出程序"<<endl;
708. cin>>flag;
709. }
710. **return** 0;
711. }

数据文件Data.txt

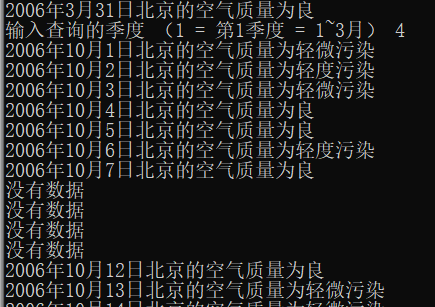
作业文件夹自带

1. 运行结果（程序运行时的结果说明或运行截图等）

普通查询： 分别查询了北京市在 第1天、第2周、第3月、第4季度的信息。注意到在查询中可能有些日期是没有数据的，比如第4季度中没有10月8日到10月11日的数据就显示的是没有数据。

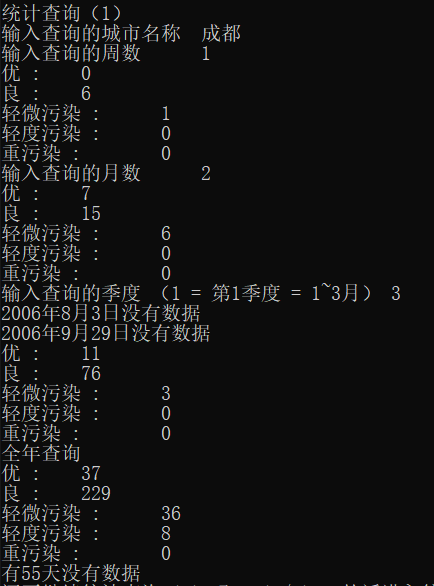


没有天气信息的会给出说明。



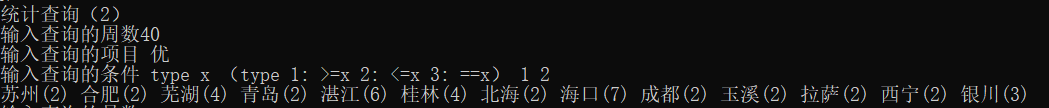
统计查询（1）：

这次分别查询了成都在第1周、第2月、第3季度、全年的天气统计信息。注意在查询信息的时候如果没有信息的话，比如8月3日和9月29日会给出提示信息，可以查看第3季度的查询。



统计查询（2）：

这一次查询的是在第40周中天气为优的天数>=2的城市有哪些。在城市名后面括号中的数字就是当前城市在第40周天气为优的天数。

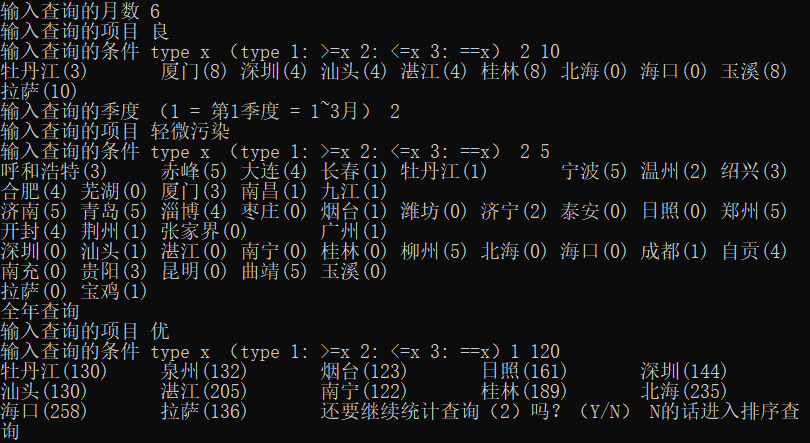


然后分别查询了

1、在第6月中天气为良的天数<=10的城市；

2、在第2季度中天气为轻微污染的天数<=5的城市；

3、在全年中天气为优的天数>=120的城市。



排序查询：

分别查询了在第一周、第12月、第2季度、全年天气平均得分最高的前5、3、5、5位。城市名后面括号里的数字是当前城市的平均天气分数。



六、总结（实验中遇到的问题、取得的经验、感想等）

1、使用散列函数大大加快了查询的速度。一开始是使用用顺序简单搜索，在搜索单个城市单天信息的时候还行，可是在后来搜索周、月的信息的时候就很慢。分析下来的原因是我的写法相当于把周查找分解成7天查找、把月查找分解成月查找，这样的话就很慢了。后来改用散列函数去插入信息，一下子速度就得到了提升，因为在大多数的查找情况下，都是直接就找到了目标信息的下标。

2、使用散列函数可以灵活的处理不同的查询任务。如果只是简单的查询城市的某一天天气的单次查询，那么把当前城市编号、日期传给f散列函数就可以得到当前城市、日期天气信息在数据数组D中的下标了，这个过程的复杂度是O(1)的，而简单搜索的话是O(n)的，主要原因就是找下标的速度不同。现在来看一些复杂的情况：

（1）、时间上的复杂，如果查询单个城市的某一月天气情况，那么把一个月拆解成30次左右的单次查询就可以了。这样的复杂度是O(30)=O(1)，而简单搜索是O(30n)的。

（2）、城市上的复杂，如果查询的是多个城市的某一天的天气情况，那么还是把城市群拆解成单个的城市去查询就好了。

（3）、城市时间上的查询：如果是遍历一遍所有信息的话，那么会查询共27263次，总的访问D数组也不会超过把数据存一遍的访问次数（1万次左右），所以非常的快。而简单搜索就是O(n^2)，差不多八千万次，就很慢。

总结的说就是借助散列函数的帮助，我们能把单次查询的速度提到了极致（O(1)）,所以再进行一些重复操作的话也会很快。

3、散列函数有局限。最大的局限就在于可能有地址冲突的情况，比如如果散列函数选的不好，只用了城市编号作为参数没用日期的话，那就会出现一个问题：过度依赖线性勘察法解决地址冲突的情况。在只用当前城市编号作为参数的情况下，单次查询的速度是大于O(170)的，比我的散列函数慢了100倍多。出现这样的速度原因是每个城市的365天的信息都挤在了一个下标中，于是，f函数平均需要找（1+2+3+…+365）/365 ≈ 170次左右才能找到，这还没考虑两个编号相近城市之间的冲突。不光这种情况下会这样，只选日期作为参数、散列函数中mod值取的是合数、mod值取得比较小等等情况下，都会使地址冲突的问题加重。虽然大多数时候不能避免地址冲突的问题，但是，我们可以控制散列函数，来减小这个问题。比如：考虑多个参数、选取较大的mod值、mod取素数、倍增参数的系数、使用更好的方法解决地址冲突等等。拿我的散列函数举例：我第一次写的时候就是将城市编号加上日期序数，后来发现所有城市的编号都是以0x结尾，所以如果将城市编号\*8 后任意两城市间编号差都 差不多>800，并且严格>400，再加上日期序数后，之前编号相近城市间的交区间就大大减少了。还有就是数据集容量的选择，因为一般的数据集就是数据量的2倍到5倍左右，所以我取了数据量的四倍作为数据集容量。然后mod值就取比数据集容量小的最大的素数。这就是我写的散列函数的思路了。顺带一提，有一些散列函数的雷区不能碰，比如这个题目中就不能用空气质量分数作为参数，因为如果现在要找北京的第一天的数据，当前是不知道当天的空气质量分数的，所以就没办法用散列函数找下标（因为当前散列函数需要空气质量），这本来是个简单的逻辑，但是第一次遇到的时候还是觉得挺有意思的。

4. 统计查询的处理：统计=查询+数据处理。利用简单查询的办法先收集信息到统计数组中去，然后再进行数据处理就好了。过程比较繁琐，但是思路很简单。

5. 排序查询，其实也是统计查询的一种。利用简单查询的办法收集信息中的分数，再进行数据处理，处理出每个城市的平均得分。最后调用sort函数对城市进行排序。

6. 可以改进的地方：我觉得这一题可以通过修改散列函数来使得单次查询的速度是1，也就是说不会出现地址冲突的情况。初步的想法就是先把77个城市新编号为1~77，然后散列函数写为：

f(城市，日期) = 城市新编号 \* 400 + 日期序数

因为不同城市间的散列值总是存在20左右的差距，所以不会出现地址冲突的情况。值得一提的是这样的数据集容量是差不多1倍的，也就是说数据集被填满了。