20220503-算法

1.过程描述

1.1

1）优化算法的标准技术

- **IO与函数调用**：使I/O减到最少，减少函数调用的次数，限制计算密集型操作（浮点数运算和除法运算）
- **关键操作**：确定执行得最频繁的算法元素，比如冒泡排序的比较和交换；
- **疏忽**：检查可能由于疏忽导致而导致特别缓慢的的实现。这往往与查找最坏情况相似

> 一个用于测试最佳缓冲区大小的程序

C++ | 复制代码

```cpp
#include <stdio.h>
#include <Windows.h>
#include <stdlib.h>
#include <limits.h>
#undef CopyFile
#pragma comment(lib,"kernel32.lib")
#define DEF_BUF 512

long GetCPUTime()
{
    static LARGE_INTEGER li = { 0 };
    //LARGE_INTEGER是一个共用体，其中的QuadPart的类型为longlong（8字节，64位）
    LARGE_INTEGER linow = { 0 };

    if (0 == li.QuadPart)
    {
        QueryPerformanceFrequency(&li);
        //输入为A pointer to a variable that receives the current performance-counter frequency, in counts per second
        //作用是返回硬件支持的高精度计数器的频率；如果为零，则表示硬件不支持，读取失败
    }
    QueryPerformanceCounter(&linow);//该函数的输入为A pointer to a variable that receives the current performance-counter value, in counts.
    return linow.QuadPart * 1000 / li.QuadPart;//利用计数与频率得出时间，以毫秒为单位(linow为总的计数，li为每毫秒的计数)
}

#define get_clock_ticks(x)\
x=GetCPUTime()

long CopyFile(char* inFileName, char* outFileName, size_t insize, size_t outsize)
{
    int c;
    long starttime, donetime;
    FILE* infile = NULL, * outfile = NULL;
    if ((infile = fopen(inFileName, "rb")) == NULL)
    {
        printf("Can't open %s\n", inFileName);
        exit(1);
    }
    if (setvbuf(infile, NULL, _IOFBF, insize))
        //setvbuf定义流如何缓冲（缓冲区大小、模式等），_IOFBF表示文件缓冲模式为全缓冲（对于输出，数据在缓冲填满时被一次性写入。对于输入，缓冲会在请求输入且缓冲为空时被
```

```
    填充)
40          //如果成功，返回0，否则返回非零值
41      {
42          printf("couldn't set infile buffer to %u bytes\n", insize);
43          exit(1);
44      }
45      if ((outfile = fopen(outFileName, "wb")) == NULL)
46      {
47          printf("can't open %s\n", outFileName);
48          exit(1);
49      }
50      if (setvbuf(outfile, NULL, _IOFBF, outsize))
51      {
52          printf("couldn't set outfile buffer to %u bytes\n", outsize);
53          exit(1);
54      }
55
56      /* do it */
57      get_clock_ticks(starttime);
58      while ((c = fgetc(infile)) != EOF)//将文件中的字符读取到流中
59          fputc(c, outfile);//把流中的字符写入到文件中
60      get_clock_ticks(donetime);
61      fclose(infile);
62      fclose(outfile);
63      return(donetime - starttime);
64  }
65
66  int main(int argc, char* argv[])
67  {
68      size_t insize, outsize;
69      int i;
70      long total, average, lo, hi, elapsed;
71      insize = outsize = DEF_BUF;
72      if (argc < 3 || argc>5)
73      {
74          fprintf(stderr,
75              "Usage: BUFSIZE infile outfile[insize[outsize]]\n");
76          return (EXIT_FAILURE);
77      }
78
79      /* get buffer sizes */
80      if (argc > 3)
81      {
82          insize = (unsigned)atoi(argv[3]);//将字符串转换为整型
83      }
84      if (argc > 4)
85      {
86          outsize = (unsigned)atoi(argv[4]);
```

```
87        }
88
89        /* now copy the file five times */
90        total = hi = 0;
91        lo = LONG_MAX;
92        for (i = 1;; i++)
93        {
94            elapsed = CopyFile(argv[1], argv[2], insize, outsize);//返回复制文
      件的用时
95            if (elapsed > hi)
96                hi = elapsed;
97            if (elapsed < lo)
98                lo = elapsed;
99            total += elapsed;//最终返回总用时
100           if (total > 500 || i > 4)
101               break;
102       }
103       average = total/i;
104       printf("Average of %d ticks (%d-%d).Insize=%u.
      Outsize=%u.\n",average, lo, hi, insize, outsize);
105       return(EXIT_SUCCESS);
106   }
107
```

2）链表

> 为城市–温度数据建立链表容器，按温度从低到高进行排列，并识别出其中的中值

C++ | 复制代码

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* data definition */
struct Node
{
    char* City;
    int Temp;
    struct Node* Next;
};
typedef struct Node* Link;
Link Head;
int NodeCount;

/* functions declarations for linked lists */
int AddNodeAscend(Link);
void CreateList();
int DeleteNode(Link);
int DuplicateNode(Link, Link);
void FreeNode(Link);
int NodeCmp(Link, Link);
/* functions definitions */
int AddNodeAscend(Link to_add)
{
    Link pn, //将要添加的节点的本地copy
        prev, //指向前一个节点
        curr; //指向正在检查的节点
    struct Node dummy;
    int i;
    /* 拷贝输入节点 */
    pn = (Link)malloc(sizeof(struct Node));
    if (pn == NULL)
        return 0;
    memcpy(pn, to_add, sizeof(struct Node));
    /* 生成一个Dummy节点来简化逻辑 */
    dummy.Next = Head;
    prev = &dummy;
    curr = Head;
    //dummy(prev)->Head(curr)

    /* 插入节点pn */
    for (;; prev = curr, curr = curr->Next)
    {
        if (curr == NULL)
```

```
46              break;
47          i = NodeCmp(pn, curr);
48          if (i <= 0)//pn的温度低于curr 或 pn字符串小于curr
49              break;
50      }
51      if (curr && i == 0)
52          if (DuplicateNode(curr, pn) == 0)
53              return (1);
54      prev -> Next = pn;
55      pn->Next = curr;
56      Head = dummy.Next;
57      //dummy的next为pn，Head变为了pn，原先Head后面的节点curr变为了pn的next
58      return (1);
59  }
60
61  int DuplicateNode(Link inlist, Link duplicate)
62  {
63      FreeNode(duplicate);
64      return (0);
65  }
66
67  int DeleteNode(Link to_delete)
68  {
69      Link curr, prev;
70      int i;
71      if (Head == NULL)
72          return(0);
73      for (prev = NULL, curr = Head; curr != NULL && (i =
    NodeCmp(to_delete, curr)) > 0; prev = curr, curr = curr->Next);
74      if (curr != NULL && i == 0)
75      {
76          if (prev)
77              prev->Next = curr->Next;
78          else
79              Head = curr->Next;
80          FreeNode(curr);
81          NodeCount -= 1;
82          return(1);
83      }
84      return (0);
85  }
86
87  int NodeCmp(Link a, Link b)
88  {
89      if (a->Temp != b->Temp)
90          return(a->Temp - b->Temp);
91      return strcmp(a->City, b->City);
92      //如果温度相等，则对两个城市的字符串长度进行比较
```

```
93          //当a的字符串大于b时，返回值大于0
94      }
95
96      void CreateList()
97      {
98          Head = NULL;
99          NodeCount = 0;
100     }
101
102     void FreeNode(Link n)
103     {
104         free(n->City);
105         free(n);
106     }
107
108     void ShowNode()
109     {
110         Link pn;
111         int count, median;
112         for (count = 0, pn = Head; pn; pn = pn->Next)
113             count += 1;
114         median = count / 2 + 1;
115         if (count)
116         {
117             count = 0;
118             for (pn = Head; pn; pn = pn->Next)
119             {
120                 printf("%-20s: %3d", pn->City, pn->Temp);
121                 count += 1;
122                 if (count == median)
123                     printf(" --Median--");
124                 printf("\n");
125             }
126         }
127         else
128             printf("Empty list\n");
129     }
130
131     int main(int argc, char* argv[])
132     {
133         FILE* fin;
134         char buffer[128];
135         struct Node n;
136         if (argc != 2)
137         {
138             fprintf(stderr, "Usage:citytemp filename.ext\n");
139             exit(EXIT_FAILURE);
140         }
```

```
141         fin = fopen(argv[1], "rt");
142         if (fin == NULL)
143         {
144             fprintf(stderr, "Cannot open/find %s\n", argv[2]);
145             exit(EXIT_FAILURE);
146         }
147         CreateList();
148         /* main loop */
149         while (!feof(fin))
150         {
151             /* 读取流中的一行 */
152             if (fgets(buffer, 127, fin) == NULL)
153                 break;
154             /* get rid of the trailing carriage return */
155             buffer[strlen(buffer) - 1] = (char)"\0";
156             /* copy the city name to the node to be added */
157             n.City = strdup(buffer + 3);
158             /* mark off the temperature and convert to int */
159             buffer[3] = (char)"\0";
160             n.Temp = atoi(buffer);
161             if (AddNodeAscend(&n) == 0)
162             {
163                 fprintf(stderr, "Error adding node. Aborting\n");
164                 exit(EXIT_FAILURE);
165             }
166         }
167         ShowNode();
168         printf("\n");
169         DeleteNode(Head);
170         ShowNode();
171         while (Head && Head->Next)
172         {
173             printf("\n");
174             DeleteNode(Head->Next);
175             ShowNode();
176         }
177         printf("\n");
178         DeleteNode(Head);
179         ShowNode();
180         fclose(fin);
181         return (EXIT_SUCCESS);
182     }
```

2.结果输出

今天只在上午跟下午看了算法的部分知识，晚上白给。五一假期也快结束了，要慢慢找回状态。比较有趣的新闻是今天微博上突然出现一个杭州马某某涉嫌分裂国家罪被捕的消息，很多人都以为是杰克马，结果后来发现是乌龙一场，为马某某而非马某。值得玩味的是，很多官媒在报道时有刻意引导吃瓜群众瞎想那味，属实令人反感。另外，有位网友说的挺有意思的：重要的不是是不是Jack Ma，而是大家真的相信他干得出来。哈哈哈哈。