当前文件夹下含有4个文件，其中test.cc为本次项目的代码，其余为测试用代码和生成文件。

主要代码如下：

（1）merge接口用于将多个文件同时合并，将各排序好的文件的文件指针所指向的内容进行比对，将最小的写入数组t中，当写满t时，能使用的内存已满，此时将t中的内容写入到文本中，然后将数据继续写入t中，可以视t为循环数组。

|  |
| --- |
| void merge(vector<string> vec, ofstream &ofs, int op){  vector<ifstream> ifs(vec.size());  for(int i = 0; i < vec.size(); i++){  ifs[i].open(vec[i].c\_str(), ios::in | ios::binary);  }  int \*t = new int[maxn << 1];  int len = 0;  int mn\_pos = 0, mn\_val;  vector<int> val(vec.size());  for(int i = 0; i < ifs.size(); i++){  if(ifs[i].is\_open()){  r\_val(ifs[i], val[i]);  mn\_val = val[i];  mn\_pos = i;  }  }  int now = 0;  while(1){  mn\_val = (1LL << 31) - 1;  int flag = 0;  for(int i = 0; i < ifs.size(); i++){  if(ifs[i].is\_open()){  if(mn\_val >= val[i]){  mn\_val = val[i];  mn\_pos = i;  flag = 1;  }  }  }  if(!flag)break;  w\_in\_t(ofs, t, len, val[mn\_pos], op, 0);  if(ifs[mn\_pos].peek() == EOF){  ifs[mn\_pos].close();  continue;  }  r\_val(ifs[mn\_pos], val[mn\_pos]);  now++;  }  w\_in\_t(ofs, t, len, 0, op, 1);  for(int i = 0; i < ifs.size(); i++){  if(ifs[i].is\_open())ifs[i].close();  remove(vec[i].c\_str());  }  ofs.close();  delete []t;  } |

（2）mem\_merge将16gb大小的文件分割为100mb每个的小文件，此时保证进程运行数量上限，可以控制使用的内存大小，因此所有进程总和不超过内存的前提下，将小文件分别进行归并排序，用于后续对文件合并同时进行排序。

|  |
| --- |
| void mem\_merge(int l, int r,int mid, int a[], int tmp[]){  int lp = l, rp = mid+1, st = l;  while(lp <= mid && rp <= r){  if(a[lp] <= a[rp]){  tmp[st++] = a[lp++];  }  else{  tmp[st++] = a[rp++];  }  }  while(lp <= mid){  tmp[st++] = a[lp++];  }  while(rp <= r){  tmp[st++] = a[rp++];  }  for(int i = l; i <= r; i++){  a[i] = tmp[i];  }  } |

本次项目的运行时间为如图：



此结果经过多次测试，误差大约为±10s，由于clock()函数在进程挂起时不会进行计算，故使用计算时间更准确的chrono类进行代码运行时间的计算。且利用当前文件夹中的生成数据的代码生成较小的数据用于检查排序的正确性。

存在以下几点可优化的空间：

①大文件的读入，由于源数据文件是以文本格式存储，导致以个人的能力只能一个一个数据进行读入，如果改为二进制，则可以按位一次性读入，加快数据读入的时间，使得分割文件时，能够跑到进程上限。

当前分割文件进程未能跑到上限的主要原因是文件读入太慢，文件分割的速度比文件读入要快，虽然CPU跑满，但内存也只能跑到上限的20%~30%。

②在最后对多个文件统合成最终结果时，以本人的能力只能进行单线程处理，如果能够对文件写入进行加锁，并且线程按创建顺序获得锁，可以使多个文件处理好的数据交给线程，等待当前文件写入的锁解锁。此时可以文件读入处理不受文件写入的处理的阻塞。