Assignment

Page No.
Date

Title: Parallel Sorting Algorithms

Problem statement: Write a program to implement Parallel Bubble sort and merge sort using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.

Objectives:

i) To implement parallel bubble sort.
ii) To implement parallel merge sort.
iii) To measure and compare performance of sequential and parallel sorting algorithms.

Outcomes:

Analysis and Results of performance of parallel sorting Algorithms.

Software / Hardware Requirements

4 GB RAM, 500 GB HDD, PC, intel i5
c++ programming, Ubuntu

Theory:

Bubble sort

It is simple sorting algorithm that works by repeate swapping adjacent elements if they are in wrong order. It is called bubble sort because algorit moves larger elements towards the end of array manner that resembles the rising of bubbles liquid.

Page No.
Date

Pseudo-code:

1. Start at the beginning of the array.
2. Compare first two elements. If first element is greater than second element, swap them.
3. Move to next pair of elements and repeat step 2.
4. Continue process until end of array is reached.
5. If any swaps were made in step 2-4, repeat process from step 1.

Time complexity of bubble sort is $O(n^2)$.

Parallel bubble sort

1. Parallel bubble sort is modification of classic bubble sort algorithm that takes advantages of parallel processing to speed up sorting process.

2. In parallel bubble sort, list of elements is divided into multiple sublists that are sorted concurrently by multiple threads. Each thread sorts in sublist using regular bubble sort algorithm. When all sublists have been sorted, they are merged together to form final sorted list.

3. The parallelization of algorithm is achieved using OpenMp, programming API that supports parallel processing in C++. OpenMp provides set of compiler directives that allow developers to specify which part of code can be executed in parallel.

Page No.
Date

4. In parallel bubble sort Algorithm, main loop that iterates over list of elements is divided into multiple iterations that are executed concurrently by multiple threads. Each thread sorts a subset of list, threads synchronize their work at end of each iteration to ensure that elements are properly ordered.

5. Parallel bubble sort can provide a significant speedup over the regular bubble sort algorithm, especially when sorting large datasets on multi core processors. However, speedup is limited by the overhead of thread execution and synchronization and it may not be worth effort for small data or when using single-core processor.

Merge sort

Merge sort is sorting algorithm that uses a divide and conquer approach to sort an array or list of elements. The algorithm works by recurssively dividing input array into two halves, sorting each half, and merging sorted halves to produce a sorted output.

The merge sort can be broken into following steps

1. Divide input array into two halves
2. Recurssively sort left half of array
3. Recurssively sort right half of array
4. Merge two sorted halves into singk sorted output array.

The merging step is bulk of work happens in merge sort. The algorithm compares the first element of each sorted half, selects smaller elements and append it to output array. This process continues until all elements from both halves have been appended to output array.
Time complexity of merge sort is O(nlogn)

Parallel Merge sort

The parallel merge sort can be broken into following steps:

1) Divide input array into smaller subarrays
2) Assign each subarray to a seperate processor or core for sorting.
3) Sort each subarray in parallel.
4) Merge the sorted subarrays together in parallel to produce final sorted output.

Parallel merge sort can provide significant performance benefits for large input array with many elements, especially when running on hardware with multiple cores. However it also requires additional overhead to manage parallelization, and may not always provide performance improvement for small inputs.

OpenMp
Open Multi-processing is API that supports shared memory parallel programming in C/C++.

It is used to write parallel programs that run on multicore processors.

OpenMp provides set of directives and functions that can be inserted into source code of program to parallelize its execution.
These directives can be applied to loops, sections, functions and other program constructs.

Conclusion:

Thus, we successfully implemented Bubble sort and Merge sort algorithm in parallel as well as sequential way and measures their performances.

Code

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

void merge(int arr[], int p, int q, int r) {

   int n1 = q - p + 1;
   int n2 = r - q;

   int L[n1], M[n2];

   for (int i = 0; i < n1; i++)
      L[i] = arr[p + i];
   for (int j = 0; j < n2; j++)
      M[j] = arr[q + 1 + j];

   int i, j, k;
   i = 0;
   j = 0;
   k = p;


   while (i < n1 && j < n2) {
      if (L[i] <= M[j]) {
         arr[k] = L[i];
         i++;
      }
      else {
         arr[k] = M[j];
```

```
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = M[j];
        j++;
        k++;
    }
}


void mergeSort(int arr[], int l, int r) {
    if (l < r) {

        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);


        merge(arr, l, m, r);
    }
}
```

```
void merge_para(int arr[],int l,int r){
    if(l<r){
        int m=l+(r-l)/2;
        #pragma omp parallel sections
        {
        #pragma omp section
        {
        merge_para(arr,l,m);
        }
        #pragma omp section
        {
        merge_para(arr,m+1,r);
        }
        }
        merge(arr,l,m,r);
    }

}
void bubbleSort(int arr[], int n)
{

    int i, j;

    for (i = 0; i < n - 1; i++){
        for (j = 0; j < n - i - 1; j++){

            if (arr[j] > arr[j + 1]){

                swap(arr[j], arr[j + 1]);
            }
        }
```

```
    }
}
void swap_para(int *a, int *b){
    int temp=*a;
    *a=*b;
    *b=temp;
}
void bubble_para(int arr[], int n){

    int i=0, j=0;
    int f;

    for (i = 0; i < n - 1; i++){
        f=i%2;
        #pragma omp parallel for default(none), shared(arr,first,n)
        for (j = f; j < n - 1; j++){

            if (arr[j] > arr[j + 1]){

                swap_para(&arr[j], &arr[j + 1]);
            }
        }
    }
}



void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}
```

```
int main() {
    int n;
    cout<<"Enter no of elements in array:";
    cin>>n;
    int arr1[n],arr2[n],arr3[n],arr4[n];
    for(int i=0;i<n;i++){
        //cin>>arr[i];
        arr1[i]=rand()%n;
        arr2[i]=arr1[i];
        arr3[i]=arr1[i];
        arr4[i]=arr1[i];
    }
    printArray(arr1,n);
    auto start = chrono :: steady_clock :: now();



    mergeSort(arr1, 0, n - 1);
    auto end = chrono :: steady_clock :: now();

    cout << " Merge Sorted array: \n";
    printArray(arr1, n);
    chrono::duration<double,micro>fp=end-start;
    cout<<fp.count()<<" microseconds"<<endl;

    auto start1 = chrono :: steady_clock :: now();

    merge_para(arr2,0,n-1);

    auto end1 = chrono :: steady_clock :: now();
```

```
    cout << "Parallel Merge Sorted array: \n";
    printArray(arr2, n);
    chrono::duration<double,micro>fp1=end1-start1;
    cout<<fp1.count()<<" microseconds"<<endl;

    auto start2 = chrono :: steady_clock :: now();
    bubbleSort(arr3, n);
    auto end2 = chrono :: steady_clock :: now();

    cout << "Bubble Sorted array: \n";

    printArray(arr3, n);
    chrono::duration<double,micro>fp2=end2-start2;
    cout<<fp2.count()<<" microseconds"<<endl;

    auto start3 = chrono::steady_clock::now();
    bubble_para(arr4,n);
    auto end3 = chrono::steady_clock::now();

    cout << "Parallel Bubble Sorted array: \n";

    printArray(arr4, n);
    chrono::duration<double,micro>ft3=end3-start3;
    cout<<ft3.count()<<" microseconds"<<endl;

    return 0;
}
```

output:

Enter no of elements in array:1000

41 467 334 500 169 724 478 358 962 464 705 145 281 827 961 491 995 942 827 436 391 604 902
153 292 382 421 716 718 895 447 726 771 538 869 912 667 299 35 894 703 811 322 333 673 664
141 711 253 868 547 644 662 757 37 859 723 741 529 778 316 35 190 842 288 106 40 942 264 648
446 805 890 729 370 350 6 101 393 548 629 623 84 954 756 840 966 376 931 308 944 439 626 323
537 538 118 82 929 541 833 115 639 658 704 930 977 306 673 386 21 745 924 72 270 829 777 573
97 512 986 290 161 636 355 767 655 574 31 52 350 150 941 724 966 430 107 191 7 337 457 287 753
383 945 909 209 758 221 588 422 946 506 30 413 168 900 591 762 655 410 359 624 537 548 483
595 41 602 350 291 836 374 20 596 21 348 199 668 484 281 734 53 999 418 938 900 788 127 467
728 893 648 483 807 421 310 617 813 514 309 616 935 451 600 249 519 556 798 303 224 8 844 609
989 702 195 485 93 343 523 587 314 503 448 200 458 618 580 796 798 281 589 798 9 157 472 622
538 292 38 179 190 657 958 191 815 888 156 511 202 634 272 55 328 646 362 886 875 433 869 142
844 416 881 998 322 651 21 699 557 476 892 389 75 712 600 510 3 869 861 688 401 789 255 423 2
585 182 285 88 426 617 757 832 932 169 154 721 189 976 329 368 692 425 555 434 549 441 512
145 60 718 753 139 423 279 996 687 529 549 437 866 949 193 195 297 416 286 105 488 282 455
734 114 701 316 671 786 263 313 355 185 53 912 808 832 945 313 756 321 558 646 982 481 144
196 222 129 161 535 450 173 466 44 659 292 439 253 24 154 510 745 649 186 313 474 22 168 18
787 905 958 391 202 625 477 414 314 824 334 874 372 159 833 70 487 297 518 177 773 270 763
668 192 985 102 480 213 627 802 99 527 625 543 924 23 972 61 181 3 432 505 593 725 31 492 142
222 286 64 900 187 360 413 974 270 170 235 833 711 760 896 667 285 550 140 694 695 624 19 125
576 694 658 302 371 466 678 593 851 484 18 464 119 152 800 87 60 926 10 757 170 315 576 227 43
758 164 109 882 86 565 487 577 474 625 627 629 928 423 520 902 962 123 596 737 261 195 525
264 260 202 116 30 326 11 771 411 547 153 520 790 924 188 763 940 851 662 829 900 713 958 578
365 7 477 200 58 439 303 760 357 324 477 108 113 887 801 850 460 428 993 384 405 540 111 704
835 356 72 350 823 485 556 216 626 357 526 357 337 271 869 361 896 22 617 112 717 696 585 41
423 129 229 565 559 932 296 855 53 962 584 734 654 972 457 369 532 963 607 483 911 635 67 848
675 938 223 142 754 511 741 175 459 825 221 870 626 934 205 783 850 398 279 701 193 734 637
534 556 993 176 705 962 548 881 300 413 641 855 855 142 462 611 877 424 678 752 443 296 673
40 313 875 72 818 610 17 932 112 695 169 831 40 488 685 90 497 589 990 145 353 314 651 740 44
258 335 759 192 605 264 181 503 829 775 608 292 997 549 556 561 627 467 541 129 240 813 174
601 77 215 683 213 992 824 601 392 759 670 428 27 84 75 786 498 970 287 847 604 503 221 663
706 363 10 171 489 240 164 542 619 913 591 704 818 232 750 205 975 539 303 422 98 247 584 648
971 864 913 75 545 712 546 678 769 262 519 985 289 944 865 540 245 508 318 870 601 323 132
472 152 87 570 763 901 103 423 527 600 969 15 565 28 543 347 88 943 637 409 463 49 681 588 342
608 60 221 758 954 888 146 690 949 843 430 620 748 67 536 783 35 226 185 38 853 629 224 748
923 359 257 766 944 955 318 726 411 25 355 1 549 496 584 515 964 342 75 913 142 196 948 72 426
606 173 429 404 705 626 812 375 93 565 36 736 141 814 994 256 652 936 838 482 355 15 131 230
841 625 11 637 186 690 650 662 634 893 353 416 452 8 262 233 454 303 634 303 256 148 124 317
213 109 28 200 80 318 858 50 155 361 264 903 676 643 909 902 561 489 948 282 653 674 220 402
923 831 369 878 259 8 619 971 3 945 781 504 392 685 313 698 589 722 938 37 410 461 234 508 961
959 493 515 269 937 869 58 700 971 264 117 215 555 815 330 39 212 288 82 954 85 710 484 774
380 815 951 541 115 679 110 898 73 788 977 132 956 689 113 8 941 790 723 363 28 184 778 200 71
885 974 71 333 867 153 295 168 825 676 629 650 598 309 693 686 80 116 249

Merge Sorted array:

1 2 3 3 3 6 7 7 8 8 8 8 9 10 10 11 11 15 15 17 18 18 19 20 21 21 21 22 22 23 24 25 27 28 28 28 30 30
31 31 35 35 35 36 37 37 38 38 39 40 40 40 41 41 41 43 44 44 49 50 52 53 53 53 55 58 58 60 60 60 61

64 67 67 70 71 71 72 72 72 72 73 75 75 75 75 77 80 80 82 82 84 84 85 86 87 87 88 88 90 93 93 97 98
99 101 102 103 105 106 107 108 109 109 110 111 112 112 113 113 114 115 115 116 116 117 118
119 123 124 125 127 129 129 129 131 132 132 139 140 141 141 142 142 142 142 142 144 145 145
145 146 148 150 152 152 153 153 153 154 154 155 156 157 159 161 161 164 164 168 168 168 169
169 169 170 170 171 173 173 174 175 176 177 179 181 181 182 184 185 185 186 186 187 188 189
190 190 191 191 192 192 193 193 195 195 195 196 196 199 200 200 200 200 202 202 202 205 205
209 212 213 213 213 215 215 216 220 221 221 221 221 222 222 223 224 224 226 227 229 230 232
233 234 235 240 240 245 247 249 249 253 253 255 256 256 257 258 259 260 261 262 262 263 264
264 264 264 264 269 270 270 270 271 272 279 279 281 281 281 282 282 285 285 286 286 287 287
288 288 289 290 291 292 292 292 292 295 296 296 297 297 299 300 302 303 303 303 303 303 306
308 309 309 310 313 313 313 313 313 314 314 314 315 316 316 317 318 318 318 321 322 322 323
323 324 326 328 329 330 333 333 334 334 335 337 337 342 342 343 347 348 350 350 350 350 353
353 355 355 355 355 356 357 357 357 358 359 359 360 361 361 362 363 363 365 368 369 369 370
371 372 374 375 376 380 382 383 384 386 389 391 391 392 392 393 398 401 402 404 405 409 410
410 411 411 413 413 413 414 416 416 416 418 421 421 422 422 423 423 423 423 423 424 425 426
426 428 428 429 430 430 432 433 434 436 437 439 439 439 441 443 446 447 448 450 451 452 454
455 457 457 458 459 460 461 462 463 464 464 466 466 467 467 467 472 472 474 474 476 477 477
477 478 480 481 482 483 483 483 484 484 484 485 485 487 487 488 488 489 489 491 492 493 496
497 498 500 503 503 503 504 505 506 508 508 510 510 511 511 512 512 514 515 515 518 519 519
520 520 523 525 526 527 527 529 529 532 534 535 536 537 537 538 538 538 539 540 540 541 541
541 542 543 543 545 546 547 547 548 548 548 549 549 549 549 550 555 555 556 556 556 556 557
558 559 561 561 565 565 565 565 570 573 574 576 576 577 578 580 584 584 584 585 585 587 588
588 589 589 589 591 591 593 593 595 596 596 598 600 600 600 601 601 601 602 604 604 605 606
607 608 608 609 610 611 616 617 617 617 618 619 619 620 622 623 624 624 625 625 625 625 626
626 626 626 627 627 627 629 629 629 629 634 634 634 635 636 637 637 637 639 641 643 644 646
646 648 648 648 649 650 650 651 651 652 653 654 655 655 657 658 658 659 662 662 662 663 664
667 667 668 668 670 671 673 673 673 674 675 676 676 678 678 678 679 681 683 685 685 686 687
688 689 690 690 692 693 694 694 695 695 696 698 699 700 701 701 702 703 704 704 704 705 705
705 706 710 711 711 712 712 713 716 717 718 718 721 722 723 723 724 724 725 726 726 728 729
734 734 734 734 736 737 740 741 741 745 745 748 748 750 752 753 753 754 756 756 757 757 757
758 758 758 759 759 760 760 762 763 763 763 766 767 769 771 771 773 774 775 777 778 778 781
783 783 786 786 787 788 788 789 790 790 796 798 798 798 800 801 802 805 807 808 811 812 813
813 814 815 815 815 818 818 823 824 824 825 825 827 827 829 829 829 831 831 832 832 833 833
833 835 836 838 840 841 842 843 844 844 847 848 850 850 851 851 853 855 855 855 858 859 861
864 865 866 867 868 869 869 869 869 869 870 870 874 875 875 877 878 881 881 882 885 886 887
888 888 890 892 893 893 894 895 896 896 898 900 900 900 900 901 902 902 902 903 905 909 909
911 912 912 913 913 913 923 923 924 924 924 926 928 929 930 931 932 932 932 934 935 936 937
938 938 938 940 941 941 942 942 943 944 944 944 945 945 945 946 948 948 949 949 951 954 954
954 955 956 958 958 958 959 961 961 962 962 962 962 963 964 966 966 969 970 971 971 971 972
972 974 974 975 976 977 977 982 985 985 986 989 990 992 993 993 994 995 996 997 998 999

101.3 microseconds

Parallel Merge Sorted array:

1 2 3 3 3 6 7 7 8 8 8 8 9 10 10 11 11 15 15 17 18 18 19 20 21 21 21 22 22 23 24 25 27 28 28 28 30 30
31 31 35 35 35 36 37 37 38 38 39 40 40 40 41 41 41 43 44 44 49 50 52 53 53 53 55 58 58 60 60 60 61
64 67 67 70 71 71 72 72 72 72 73 75 75 75 75 77 80 80 82 82 84 84 85 86 87 87 88 88 90 93 93 97 98
99 101 102 103 105 106 107 108 109 109 110 111 112 112 113 113 114 115 115 116 116 117 118

119 123 124 125 127 129 129 129 131 132 132 139 140 141 141 142 142 142 142 142 144 145 145
145 146 148 150 152 152 153 153 153 154 154 155 156 157 159 161 161 164 164 168 168 168 169
169 169 170 170 171 173 173 174 175 176 177 179 181 181 182 184 185 185 186 186 187 188 189
190 190 191 191 192 192 193 193 195 195 195 196 196 199 200 200 200 200 202 202 202 205 205
209 212 213 213 213 215 215 216 220 221 221 221 221 222 222 223 224 224 226 227 229 230 232
233 234 235 240 240 245 247 249 249 253 253 255 256 256 257 258 259 260 261 262 262 263 264
264 264 264 264 269 270 270 270 271 272 279 279 281 281 281 282 282 285 285 286 286 287 287
288 288 289 290 291 292 292 292 292 295 296 296 297 297 299 300 302 303 303 303 303 303 306
308 309 309 310 313 313 313 313 313 314 314 314 315 316 316 317 318 318 318 321 322 322 323
323 324 326 328 329 330 333 333 334 334 335 337 337 342 342 343 347 348 350 350 350 350 353
353 355 355 355 355 356 357 357 357 358 359 359 360 361 361 362 363 363 365 368 369 369 370
371 372 374 375 376 380 382 383 384 386 389 391 391 392 392 393 398 401 402 404 405 409 410
410 411 411 413 413 413 414 416 416 416 418 421 421 422 422 423 423 423 423 423 424 425 426
426 428 428 429 430 430 432 433 434 436 437 439 439 439 441 443 446 447 448 450 451 452 454
455 457 457 458 459 460 461 462 463 464 464 466 466 467 467 467 472 472 474 474 476 477 477
477 478 480 481 482 483 483 483 484 484 484 485 485 487 487 488 488 489 489 491 492 493 496
497 498 500 503 503 503 504 505 506 508 508 510 510 511 511 512 512 514 515 515 518 519 519
520 520 523 525 526 527 527 529 529 532 534 535 536 537 537 538 538 538 539 540 540 541 541
541 542 543 543 545 546 547 547 548 548 548 549 549 549 549 550 555 555 556 556 556 556 557
558 559 561 561 565 565 565 565 570 573 574 576 576 577 578 580 584 584 584 585 585 587 588
588 589 589 589 591 591 593 593 595 596 596 598 600 600 600 601 601 601 602 604 604 605 606
607 608 608 609 610 611 616 617 617 617 618 619 619 620 622 623 624 624 625 625 625 625 626
626 626 626 627 627 627 629 629 629 629 634 634 634 635 636 637 637 637 639 641 643 644 646
646 648 648 648 649 650 650 651 651 652 653 654 655 655 657 658 658 659 662 662 662 663 664
667 667 668 668 670 671 673 673 673 674 675 676 676 678 678 678 679 681 683 685 685 686 687
688 689 690 690 692 693 694 694 695 695 696 698 699 700 701 701 702 703 704 704 704 705 705
705 706 710 711 711 712 712 713 716 717 718 718 721 722 723 723 724 724 725 726 726 728 729
734 734 734 734 736 737 740 741 741 745 745 748 748 750 752 753 753 754 756 756 757 757 757
758 758 758 759 759 760 760 762 763 763 763 766 767 769 771 771 773 774 775 777 778 778 781
783 783 786 786 787 788 788 789 790 790 796 798 798 798 800 801 802 805 807 808 811 812 813
813 814 815 815 815 818 818 823 824 824 825 825 827 827 829 829 829 831 831 832 832 833 833
833 835 836 838 840 841 842 843 844 844 847 848 850 850 851 851 853 855 855 855 858 859 861
864 865 866 867 868 869 869 869 869 869 870 870 874 875 875 877 878 881 881 882 885 886 887
888 888 890 892 893 893 894 895 896 896 898 900 900 900 900 901 902 902 902 903 905 909 909
911 912 912 913 913 913 923 923 924 924 924 926 928 929 930 931 932 932 932 934 935 936 937
938 938 938 940 941 941 942 942 943 944 944 944 945 945 945 946 948 948 949 949 951 954 954
954 955 956 958 958 958 959 961 961 962 962 962 962 963 964 966 966 969 970 971 971 971 972
972 974 974 975 976 977 977 982 985 985 986 989 990 992 993 993 994 995 996 997 998 999

102.7 microseconds

Bubble Sorted array:

1 2 3 3 3 6 7 7 8 8 8 8 9 10 10 11 11 15 15 17 18 18 19 20 21 21 21 22 22 23 24 25 27 28 28 28 30 30
31 31 35 35 35 36 37 37 38 38 39 40 40 40 41 41 41 43 44 44 49 50 52 53 53 53 55 58 58 60 60 60 61
64 67 67 70 71 71 72 72 72 72 73 75 75 75 75 77 80 80 82 82 84 84 85 86 87 87 88 88 90 93 93 97 98
99 101 102 103 105 106 107 108 109 109 110 111 112 112 113 113 114 115 115 116 116 117 118
119 123 124 125 127 129 129 129 131 132 132 139 140 141 141 142 142 142 142 142 144 145 145
145 146 148 150 152 152 153 153 153 154 154 155 156 157 159 161 161 164 164 168 168 168 169

169 169 170 170 171 173 173 174 175 176 177 179 181 181 182 184 185 185 186 186 187 188 189
190 190 191 191 192 192 193 193 195 195 195 196 196 199 200 200 200 200 202 202 202 205 205
209 212 213 213 213 215 215 216 220 221 221 221 221 222 222 223 224 224 226 227 229 230 232
233 234 235 240 240 245 247 249 249 253 253 255 256 256 257 258 259 260 261 262 262 263 264
264 264 264 264 269 270 270 270 271 272 279 279 281 281 281 282 282 285 285 286 286 287 287
288 288 289 290 291 292 292 292 292 295 296 296 297 297 299 300 302 303 303 303 303 303 306
308 309 309 310 313 313 313 313 313 314 314 314 315 316 316 317 318 318 318 321 322 322 323
323 324 326 328 329 330 333 333 334 334 335 337 337 342 342 343 347 348 350 350 350 350 353
353 355 355 355 355 356 357 357 357 358 359 359 360 361 361 362 363 363 365 368 369 369 370
371 372 374 375 376 380 382 383 384 386 389 391 391 392 392 393 398 401 402 404 405 409 410
410 411 411 413 413 413 414 416 416 416 418 421 421 422 422 423 423 423 423 423 424 425 426
426 428 428 429 430 430 432 433 434 436 437 439 439 439 441 443 446 447 448 450 451 452 454
455 457 457 458 459 460 461 462 463 464 464 466 466 467 467 467 472 472 474 474 476 477 477
477 478 480 481 482 483 483 483 484 484 484 485 485 487 487 488 488 489 489 491 492 493 496
497 498 500 503 503 503 504 505 506 508 508 510 510 511 511 512 512 514 515 515 518 519 519
520 520 523 525 526 527 527 529 529 532 534 535 536 537 537 538 538 538 539 540 540 541 541
541 542 543 543 545 546 547 547 548 548 548 549 549 549 549 550 555 555 556 556 556 556 557
558 559 561 561 565 565 565 565 570 573 574 576 576 577 578 580 584 584 584 585 585 587 588
588 589 589 589 591 591 593 593 595 596 596 598 600 600 600 601 601 601 602 604 604 605 606
607 608 608 609 610 611 616 617 617 617 618 619 619 620 622 623 624 624 625 625 625 625 626
626 626 626 627 627 627 629 629 629 629 634 634 634 635 636 637 637 637 639 641 643 644 646
646 648 648 648 649 650 650 651 651 652 653 654 655 655 657 658 658 659 662 662 662 663 664
667 667 668 668 670 671 673 673 673 674 675 676 676 678 678 678 679 681 683 685 685 686 687
688 689 690 690 692 693 694 694 695 695 696 698 699 700 701 701 702 703 704 704 704 705 705
705 706 710 711 711 712 712 713 716 717 718 718 721 722 723 723 724 724 725 726 726 728 729
734 734 734 734 736 737 740 741 741 745 745 748 748 750 752 753 753 754 756 756 757 757 757
758 758 758 759 759 760 760 762 763 763 763 766 767 769 771 771 773 774 775 777 778 778 781
783 783 786 786 787 788 788 789 790 790 796 798 798 798 800 801 802 805 807 808 811 812 813
813 814 815 815 815 818 818 823 824 824 825 825 827 827 829 829 829 831 831 832 832 833 833
833 835 836 838 840 841 842 843 844 844 847 848 850 850 851 851 853 855 855 855 858 859 861
864 865 866 867 868 869 869 869 869 869 870 870 874 875 875 877 878 881 881 882 885 886 887
888 888 890 892 893 893 894 895 896 896 898 900 900 900 900 901 902 902 902 903 905 909 909
911 912 912 913 913 913 923 923 924 924 924 926 928 929 930 931 932 932 932 934 935 936 937
938 938 938 940 941 941 942 942 943 944 944 944 945 945 945 946 948 948 949 949 951 954 954
954 955 956 958 958 958 959 961 961 962 962 962 962 963 964 966 966 969 970 971 971 971 972
972 974 974 975 976 977 977 982 985 985 986 989 990 992 993 993 994 995 996 997 998 999

2657.9 microseconds

Parallel Bubble Sorted array:

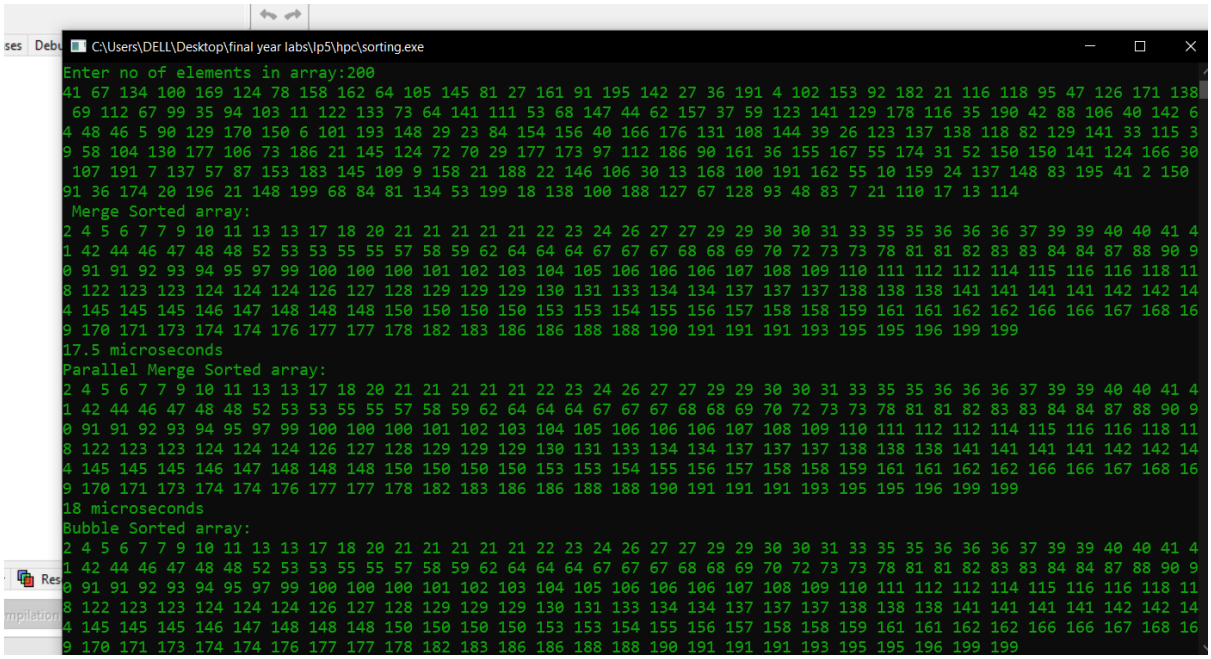
1 2 3 3 3 6 7 7 8 8 8 8 9 10 10 11 11 15 15 17 18 18 19 20 21 21 21 22 22 23 24 25 27 28 28 28 30 30
31 31 35 35 35 36 37 37 38 38 39 40 40 40 41 41 41 43 44 44 49 50 52 53 53 53 55 58 58 60 60 60 61
64 67 67 70 71 71 72 72 72 72 73 75 75 75 75 77 80 80 82 82 84 84 85 86 87 87 88 88 90 93 93 97 98
99 101 102 103 105 106 107 108 109 109 110 111 112 112 113 113 114 115 115 116 116 117 118
119 123 124 125 127 129 129 129 131 132 132 139 140 141 141 142 142 142 142 142 144 145 145
145 146 148 150 152 152 153 153 153 154 154 155 156 157 159 161 161 164 164 168 168 168 169
169 169 170 170 171 173 173 174 175 176 177 179 181 181 182 184 185 185 186 186 187 188 189
190 190 191 191 192 192 193 193 195 195 195 196 196 199 200 200 200 200 202 202 202 205 205

209 212 213 213 213 215 215 216 220 221 221 221 221 222 222 223 224 224 226 227 229 230 232
233 234 235 240 240 245 247 249 249 253 253 255 256 256 257 258 259 260 261 262 262 263 264
264 264 264 264 269 270 270 270 271 272 279 279 281 281 281 282 282 285 285 286 286 287 287
288 288 289 290 291 292 292 292 292 295 296 296 297 297 299 300 302 303 303 303 303 303 306
308 309 309 310 313 313 313 313 313 314 314 314 315 316 316 317 318 318 318 321 322 322 323
323 324 326 328 329 330 333 333 334 334 335 337 337 342 342 343 347 348 350 350 350 350 353
353 355 355 355 355 356 357 357 357 358 359 359 360 361 361 362 363 363 365 368 369 369 370
371 372 374 375 376 380 382 383 384 386 389 391 391 392 392 393 398 401 402 404 405 409 410
410 411 411 413 413 413 414 416 416 416 418 421 421 422 422 423 423 423 423 423 424 425 426
426 428 428 429 430 430 432 433 434 436 437 439 439 439 441 443 446 447 448 450 451 452 454
455 457 457 458 459 460 461 462 463 464 464 466 466 467 467 467 472 472 474 474 476 477 477
477 478 480 481 482 483 483 483 484 484 484 485 485 487 487 488 488 489 489 491 492 493 496
497 498 500 503 503 503 504 505 506 508 508 510 510 511 511 512 512 514 515 515 518 519 519
520 520 523 525 526 527 527 529 529 532 534 535 536 537 537 538 538 538 539 540 540 541 541
541 542 543 543 545 546 547 547 548 548 548 549 549 549 549 550 555 555 556 556 556 556 557
558 559 561 561 565 565 565 565 570 573 574 576 576 577 578 580 584 584 584 585 585 587 588
588 589 589 589 591 591 593 593 595 596 596 598 600 600 600 601 601 601 602 604 604 605 606
607 608 608 609 610 611 616 617 617 617 618 619 619 620 622 623 624 624 625 625 625 625 626
626 626 626 627 627 627 629 629 629 629 634 634 634 635 636 637 637 637 639 641 643 644 646
646 648 648 648 649 650 650 651 651 652 653 654 655 655 657 658 658 659 662 662 662 663 664
667 667 668 668 670 671 673 673 673 674 675 676 676 678 678 678 679 681 683 685 685 686 687
688 689 690 690 692 693 694 694 695 695 696 698 699 700 701 701 702 703 704 704 704 705 705
705 706 710 711 711 712 712 713 716 717 718 718 721 722 723 723 724 724 725 726 726 728 729
734 734 734 734 736 737 740 741 741 745 745 748 748 750 752 753 753 754 756 756 757 757 757
758 758 758 759 759 760 760 762 763 763 763 766 767 769 771 771 773 774 775 777 778 778 781
783 783 786 786 787 788 788 789 790 790 796 798 798 798 800 801 802 805 807 808 811 812 813
813 814 815 815 815 818 818 823 824 824 825 825 827 827 829 829 829 831 831 832 832 833 833
833 835 836 838 840 841 842 843 844 844 847 848 850 850 851 851 853 855 855 855 858 859 861
864 865 866 867 868 869 869 869 869 869 870 870 874 875 875 877 878 881 881 882 885 886 887
888 888 890 892 893 893 894 895 896 896 898 900 900 900 900 901 902 902 902 903 905 909 909
911 912 912 913 913 913 923 923 924 924 924 926 928 929 930 931 932 932 932 934 935 936 937
938 938 938 940 941 941 942 942 943 944 944 944 945 945 945 946 948 948 949 949 951 954 954
954 955 956 958 958 958 959 961 961 962 962 962 962 963 964 966 966 969 970 971 971 971 972
972 974 974 975 976 977 977 982 985 985 986 989 990 992 993 993 994 995 996 997 998 999

3630.5 microseconds

Process exited after 5.864 seconds with return value 0

Press any key to continue . . .


C:\Users\DELL\Desktop\final year labs\lp5\hpc\sorting.exe
Enter no of elements in array:200
41 67 134 100 169 124 78 158 162 64 105 145 81 27 161 91 195 142 27 36 191 4 102 153 92 182 21 116 118 95 47 126 171 138
69 112 67 99 35 94 103 11 122 133 73 64 141 111 53 68 147 44 62 157 37 59 123 141 129 178 116 35 190 42 88 106 40 142 6
4 48 46 5 90 129 170 150 6 101 193 148 29 23 84 154 156 40 166 176 131 108 144 39 26 123 137 138 118 82 129 141 33 115 3
9 58 104 130 177 106 73 186 21 145 124 72 70 29 177 173 97 112 186 90 161 36 155 167 55 174 31 52 150 150 141 124 166 30
107 191 7 137 57 87 153 183 145 109 9 158 21 188 22 146 106 30 13 168 100 191 162 55 10 159 24 137 148 83 195 41 2 150
91 36 174 20 196 21 148 199 68 84 81 134 53 199 18 138 100 188 127 67 128 93 48 83 7 21 110 17 13 114
Merge Sorted array:
2 4 5 6 7 7 9 10 11 13 13 17 18 20 21 21 21 21 21 22 23 24 26 27 27 29 29 30 30 31 33 35 35 36 36 36 37 39 39 40 40 41 4
1 42 44 46 47 48 48 52 53 53 55 55 57 58 59 62 64 64 64 67 67 67 68 68 69 70 72 73 73 78 81 81 82 83 83 84 84 87 88 90 9
0 91 91 92 93 94 95 97 99 100 100 100 101 102 103 104 105 106 106 106 107 108 109 110 111 112 112 114 115 116 116 118 11
8 122 123 123 124 124 124 126 127 128 129 129 129 130 131 133 134 134 137 137 137 138 138 138 141 141 141 141 142 142 14
4 145 145 145 146 147 148 148 148 150 150 150 150 153 153 154 155 156 157 158 158 159 161 161 162 162 166 166 167 168 16
9 170 171 173 174 174 176 177 177 178 182 183 186 186 188 188 190 191 191 191 193 195 195 196 199 199
17.5 microseconds
Parallel Merge Sorted array:
2 4 5 6 7 7 9 10 11 13 13 17 18 20 21 21 21 21 21 22 23 24 26 27 27 29 29 30 30 31 33 35 35 36 36 36 37 39 39 40 40 41 4
1 42 44 46 47 48 48 52 53 53 55 55 57 58 59 62 64 64 64 67 67 67 68 68 69 70 72 73 73 78 81 81 82 83 83 84 84 87 88 90 9
0 91 91 92 93 94 95 97 99 100 100 100 101 102 103 104 105 106 106 106 107 108 109 110 111 112 112 114 115 116 116 118 11
8 122 123 123 124 124 124 126 127 128 129 129 129 130 131 133 134 134 137 137 137 138 138 138 141 141 141 141 142 142 14
4 145 145 145 146 147 148 148 148 150 150 150 150 153 153 154 155 156 157 158 158 159 161 161 162 162 166 166 167 168 16
9 170 171 173 174 174 176 177 177 178 182 183 186 186 188 188 190 191 191 191 193 195 195 196 199 199
18 microseconds
Bubble Sorted array:
2 4 5 6 7 7 9 10 11 13 13 17 18 20 21 21 21 21 21 22 23 24 26 27 27 29 29 30 30 31 33 35 35 36 36 36 37 39 39 40 40 41 4
1 42 44 46 47 48 48 52 53 53 55 55 57 58 59 62 64 64 64 67 67 67 68 68 69 70 72 73 73 78 81 81 82 83 83 84 84 87 88 90 9
0 91 91 92 93 94 95 97 99 100 100 100 101 102 103 104 105 106 106 106 107 108 109 110 111 112 112 114 115 116 116 118 11
8 122 123 123 124 124 124 126 127 128 129 129 129 130 131 133 134 134 137 137 137 138 138 138 141 141 141 141 142 142 14
4 145 145 145 146 147 148 148 148 150 150 150 150 153 153 154 155 156 157 158 158 159 161 161 162 162 166 166 167 168 16
9 170 171 173 174 174 176 177 177 178 182 183 186 186 188 188 190 191 191 191 193 195 195 196 199 199