

HW2_Report 109550027 紀竺均

Part one: sort

- Introduction

- Introduction of different method/ algorithm

這部分實作了兩個 sort，Quicksort 和 Mergesort，兩者都是 Divide and conquer algorithm。Quicksort 每一輪將最右邊的點作為 pivot，反覆將 array 利用 pivot 分為左右兩邊，最後整理好整個 array；而 Mergesort 則是將 array 不斷分成兩半直到最小的單位，接著把重複把兩邊 Merge 在一起，相對於 Quicksort，Mergesort 需要創造額外的空間給被分開的 array。

- Implement Details

- Steps

- Quicksort

主要有兩個 function: quicksort()和 partition()，partition()的功能就是利用交換將 array 分成 pivot 的左右兩邊，最後 return 一個 i 值紀錄 pivot's index，而 quicksort()一直不斷遞迴的 partition 並將 array 分成 i 的兩左右邊，最後重新進行 quicksort。

- Mergesort

主要有兩個 function: mergesort()和 merge()，mergesort()裡面包含分割的概念，直到不能繼續分割後才會由小到大開始 merge；merge()先創造兩個 array 將左右兩邊傳入，最後進行比較大小。

比較大小是如果 left 比較小，就加回 A 裡面，i++，反之亦同。

Time.h 的部分，用 clock_t starttime = clock() 以及 clock_t endtime = clock() 去包住我要測量的時間，然後再用 float(endtime-starttime) / CLOCKS_PER_SEC(=1000) 就可以得到時間差的秒數。

● Results

■ Quicksort 測資數：5 時間：0

```
C:\Users\chuch\Desktop\HW2_109550027\Quicksort_109550027.exe
5
7 -1 -2 4 9
-2 -1 4 7 9
spent time: 0 seconds
7
8 4 5 -10 3 4 -6
-10 -6 3 4 4 5 8
spent time: 0 seconds
0
_
```

測資數：516 時間：0.002s

```
C:\Users\chuch\Desktop\HW2_109550027\Quicksort_109550027.exe
516
9964 9912 9810 9808 9782 9776 9771 9726 9714 9709 9695 9683 9645 9639 9590 9557 9510 9490 9484 9447 9404 9321 9312 9310 9309 9283 9266 9264 9164 9135 9108 90
76 9073 9029 8970 8938 8891 8885 8836 8764 8731 8672 8666 8614 8586 8583 8572 8560 8426 8366 8361 8356 8331 8302 8153 8125 8108 8102 8085 8028 8026 8010 7864
7852 7843 7822 7818 7755 7724 7720 7708 7647 7610 7534 7519 7481 7436 7413 7350 7341 7220 7146 7118 7072 7054 7037 7005 6978 6921 6896 6891 6890 6886 6865 6
841 6831 6813 6774 6726 6652 6638 6588 6558 6527 6429 6427 6385 6362 6355 6299 6290 6238 6232 6080 6078 6062 6008 5917 5909 5815 5769 5750 5733 5713 5668 558
5 5532 5514 5465 5439 5405 5393 5337 5335 5247 5246 5201 5169 5065 5034 5034 4997 4949 4934 4847 4786 4782 4782 4727 4523 4513 4505 4462 4418 4413 4383 4376
4366 4355 4309 4305 4284 4252 4249 4243 4189 4138 4126 4078 4057 3977 3966 3964 3893 3869 3858 3830 3809 3807 3783 3735 3594 3562 3512 3488 3426 3379 3376 33
09 3291 3290 3265 3197 3103 3099 3073 3058 3031 2911 2904 2901 2892 2818 2816 2805 2765 2723 2635 2590 2477 2404 2394 2393 2338 2306 2278 2264 2251 2196 2192
2060 2045 2029 2019 1949 1943 1919 1912 1906 1893 1857 1831 1767 1754 1753 1715 1667 1625 1609 1598 1496 1457 1446 1445 1370 1338 1304 1213 1211 1191 1141 1
128 1063 1058 1029 883 878 824 823 802 794 763 652 630 621 614 463 454 447 404 372 281 239 238 219 171 67 45 41 33 23 16 -22 -40 -101 -106 -155 -215 -222 -25
0 -262 -371 -382 -470 -526 -542 -631 -666 -715 -738 -750 -905 -975 -993 -1009 -1172 -1273 -1274 -1321 -1322 -1330 -1335 -1369 -1389 -1396 -1408 -1491 -1551 -
1615 -1644 -1646 -1709 -1713 -1760 -1777 -1790 -1812 -1843 -1852 -1905 -1933 -1968 -2162 -2245 -2266 -2267 -2270 -2294 -2297 -2398 -2465 -2624 -2670 -2704 -2
780 -2794 -2802 -2802 -2804 -2820 -2825 -2842 -2843 -2880 -2906 -3012 -3119 -3129 -3200 -3277 -3278 -3282 -3357 -3418 -3483 -3490 -3600 -3639 -3643 -3670 -36
89 -3756 -3792 -3798 -3831 -3842 -3929 -3942 -3947 -3976 -4025 -4077 -4081 -4176 -4298 -4413 -4472 -4482 -4483 -4519 -4539 -4568 -4574 -4597 -4666 -4701 -474
2 -4755 -4989 -5138 -5151 -5197 -5324 -5482 -5544 -5677 -5854 -5909 -5980 -6028 -6120 -6163 -6194 -6195 -6230 -6256 -6270 -6319 -6354 -6362 -6482 -6498 -6528
-6574 -6655 -6772 -6817 -6912 -6922 -6961 -6963 -7029 -7108 -7112 -7192 -7196 -7244 -7361 -7363 -7390 -7432 -7433 -7450 -7516 -7555 -7555 -7582 -7597 -7601
-7643 -7658 -7671 -7674 -7708 -7710 -7731 -7760 -7789 -7858 -7885 -7894 -7900 -7903 -7933 -7936 -7951 -7963 -7968 -8018 -8049 -8088 -8126 -8166 -8219 -8224 -
8304 -8331 -8481 -8486 -8511 -8538 -8629 -8650 -8733 -8835 -9142 -9150 -9225 -9251 -9253 -9284 -9323 -9331 -9336 -9393 -9431 -9501 -9505 -9510 -9515 -9559 -9
565 -9571 -9589 -9594 -9617 -9641 -9711 -9738 -9757 -9827 -9874 -9912 -9919 -9974 -9982
-9982 -9974 -9919 -9912 -9874 -9827 -9757 -9738 -9711 -9641 -9617 -9594 -9589 -9571 -9565 -9559 -9515 -9510 -9505 -9501 -9431 -9393 -9336 -9331 -9323 -9284 -
9253 -9251 -9225 -9150 -9142 -8835 -8733 -8650 -8629 -8538 -8511 -8486 -8481 -8331 -8304 -8224 -8219 -8166 -8126 -8088 -8049 -8018 -7968 -7963 -7951 -7936 -7
933 -7903 -7900 -7894 -7885 -7858 -7789 -7760 -7731 -7710 -7708 -7674 -7671 -7658 -7643 -7601 -7597 -7582 -7555 -7555 -7516 -7450 -7433 -7432 -7390 -7363 -73
61 -7244 -7196 -7192 -7112 -7108 -7029 -6963 -6961 -6922 -6912 -6817 -6772 -6655 -6574 -6528 -6498 -6482 -6362 -6354 -6319 -6270 -6256 -6230 -6195 -6194 -616
3 -6120 -6028 -5980 -5909 -5854 -5677 -5544 -5482 -5324 -5197 -5151 -5138 -4989 -4755 -4742 -4701 -4666 -4597 -4574 -4568 -4539 -4519 -4483 -4482 -4472 -4413
-4298 -4176 -4081 -4077 -4025 -3976 -3947 -3942 -3929 -3842 -3831 -3798 -3792 -3756 -3689 -3670 -3643 -3639 -3600 -3490 -3483 -3418 -3357 -3282 -3278 -3277
-3200 -3129 -3119 -3012 -2906 -2880 -2843 -2842 -2825 -2820 -2804 -2802 -2802 -2794 -2780 -2704 -2670 -2624 -2465 -2398 -2297 -2294 -2270 -2267 -2266 -2245 -
2162 -1968 -1933 -1905 -1852 -1843 -1812 -1790 -1777 -1760 -1713 -1709 -1646 -1644 -1615 -1551 -1491 -1408 -1396 -1389 -1369 -1335 -1330 -1322 -1321 -1274 -1
273 -1172 -1009 -993 -975 -905 -750 -738 -715 -666 -631 -542 -526 -470 -382 -371 -262 -250 -222 -215 -155 -106 -101 -40 -22 16 23 33 41 45 67 171 219 238 239
281 372 404 447 454 463 614 621 630 652 763 794 802 823 824 878 883 1029 1058 1063 1128 1141 1191 1211 1213 1304 1338 1370 1445 1446 1457 1496 1598 1609 162
5 1667 1715 1753 1754 1767 1831 1857 1893 1906 1912 1919 1943 1949 2019 2045 2060 2192 2196 2251 2264 2278 2306 2338 2393 2394 2404 2477 2590 2635 2723
2765 2805 2816 2818 2892 2901 2904 2911 3031 3058 3073 3099 3103 3197 3265 3290 3291 3309 3376 3379 3426 3488 3512 3562 3594 3735 3783 3807 3809 3830 3858 38
69 3893 3964 3966 3977 4057 4078 4126 4138 4189 4243 4249 4252 4284 4305 4309 4355 4366 4376 4383 4413 4418 4462 4505 4513 4523 4727 4782 4782 4786 4847 4934
4949 4997 5034 5034 5065 5169 5201 5246 5247 5335 5337 5393 5405 5439 5465 5514 5532 5585 5668 5713 5733 5750 5769 5815 5909 5917 6008 6062 6078 6080 6232 6
238 6290 6299 6355 6362 6385 6427 6429 6527 6558 6588 6638 6652 6726 6774 6813 6831 6841 6865 6886 6890 6891 6896 6921 6978 7005 7037 7054 7072 7118 7146 722
0 7341 7350 7413 7436 7481 7519 7534 7610 7647 7708 7720 7724 7755 7818 7822 7843 7852 7864 8010 8026 8028 8085 8102 8108 8125 8153 8302 8331 8356 8361 8366
8426 8560 8572 8583 8586 8614 8666 8672 8731 8764 8836 8885 8891 8938 8970 9029 9073 9076 9108 9135 9164 9264 9266 9283 9309 9310 9312 9321 9404 9447 9484 94
90 9510 9557 9590 9639 9645 9683 9695 9709 9714 9726 9771 9776 9782 9808 9810 9912 9964
spent time: 0.002 seconds
```

■ Mergesort 測資數：5 時間：0

```
C:\Users\chuch\Desktop\HW2_109550027\Mergesort_109550027.exe
5
7 -1 -2 4 9
2 -1 4 7 9
spent time: 0 seconds
7
8 4 5 -10 3 4 -6
-10 -6 3 4 4 5 8
spent time: 0 seconds
0
-----
Process exited after 1.043 seconds with return value 0
請按任意鍵繼續 . . .
```

測資數：516 時間：0

```
C:\Users\chuch\Desktop\HW2_109550027\Mergesort_109550027.exe
516
9964 9912 9810 9808 9782 9776 9771 9726 9714 9709 9695 9683 9645 9639 9590 9557 9510 9490 9484 9447 9404 9321 9312 9310 9309 9283 9266 9264 9164 9135 9108 90
76 9073 9029 8970 8938 8891 8885 8836 8764 8731 8672 8666 8614 8586 8583 8572 8560 8426 8366 8361 8356 8331 8302 8153 8125 8108 8102 8085 8028 8026 8010 7864
7852 7843 7822 7818 7755 7724 7720 7708 7647 7610 7534 7519 7481 7436 7413 7350 7341 7220 7146 7118 7072 7054 7037 7005 6978 6921 6896 6891 6890 6886 6865 6
841 6831 6813 6774 6726 6652 6638 6588 6558 6527 6429 6427 6385 6362 6355 6299 6290 6238 6232 6080 6078 6062 6008 5917 5909 5815 5769 5750 5733 5713 5668 558
5 5532 5514 5465 5439 5405 5393 5337 5335 5247 5246 5201 5169 5065 5034 5034 4997 4949 4934 4847 4786 4782 4782 4727 4523 4513 4505 4462 4418 4413 4383 4376
4366 4355 4309 4305 4284 4252 4249 4243 4189 4138 4126 4078 4057 3977 3966 3964 3893 3869 3858 3830 3809 3807 3783 3735 3594 3562 3512 3488 3426 3379 3376 33
09 3291 3290 3265 3197 3103 3099 3073 3058 3031 2911 2904 2901 2892 2818 2816 2805 2765 2723 2635 2590 2477 2404 2394 2393 2338 2306 2278 2264 2251 2196 2192
2060 2045 2029 2019 1949 1943 1919 1912 1906 1893 1857 1831 1767 1754 1753 1715 1667 1625 1609 1598 1496 1457 1446 1445 1370 1338 1304 1213 1211 1191 1141 1
128 1063 1058 1029 883 878 824 823 802 794 763 652 630 621 614 463 454 447 404 372 281 239 238 219 171 67 45 41 33 23 16 -22 -40 -101 -106 -155 -215 -222 -25
0 -262 -371 -382 -470 -526 -542 -631 -666 -715 -738 -750 -905 -975 -993 -1009 -1172 -1273 -1274 -1321 -1322 -1330 -1335 -1369 -1389 -1396 -1408 -1491 -1551 -1
1615 -1644 -1646 -1709 -1713 -1760 -1777 -1790 -1812 -1843 -1852 -1905 -1933 -1968 -2162 -2245 -2266 -2267 -2270 -2294 -2297 -2398 -2465 -2624 -2670 -2704 -2
780 -2794 -2802 -2802 -2804 -2820 -2825 -2842 -2843 -2880 -2906 -3012 -3119 -3129 -3200 -3277 -3278 -3282 -3357 -3418 -3483 -3490 -3600 -3639 -3643 -3670 -36
89 -3756 -3792 -3798 -3831 -3842 -3929 -3942 -3947 -3976 -4025 -4077 -4081 -4176 -4298 -4413 -4472 -4482 -4483 -4519 -4539 -4568 -4574 -4597 -4666 -4701 -474
2 -4755 -4989 -5138 -5151 -5197 -5324 -5482 -5544 -5677 -5854 -5909 -5980 -6028 -6120 -6163 -6194 -6195 -6230 -6256 -6270 -6319 -6354 -6362 -6482 -6498 -6528
-6574 -6655 -6772 -6817 -6912 -6922 -6961 -6963 -7029 -7108 -7112 -7192 -7196 -7244 -7361 -7363 -7390 -7432 -7433 -7450 -7516 -7555 -7555 -7582 -7597 -7601
-7643 -7658 -7671 -7674 -7708 -7710 -7731 -7760 -7789 -7858 -7885 -7894 -7900 -7903 -7933 -7936 -7951 -7963 -7968 -8018 -8049 -8088 -8126 -8166 -8219 -8224 -8
8304 -8331 -8481 -8486 -8511 -8538 -8629 -8650 -8733 -8835 -9142 -9150 -9225 -9251 -9253 -9284 -9323 -9331 -9336 -9393 -9431 -9501 -9505 -9510 -9515 -9559 -9
565 -9571 -9589 -9594 -9617 -9641 -9711 -9738 -9757 -9827 -9874 -9912 -9919 -9974 -9982
-9982 -9974 -9919 -9912 -9874 -9827 -9757 -9738 -9711 -9641 -9617 -9594 -9589 -9571 -9565 -9559 -9515 -9510 -9505 -9501 -9431 -9393 -9336 -9331 -9323 -9284 -9
9253 -9251 -9225 -9150 -9142 -8835 -8733 -8650 -8629 -8538 -8511 -8486 -8481 -8331 -8304 -8224 -8219 -8166 -8126 -8088 -8049 -8018 -7968 -7963 -7951 -7936 -7
933 -7903 -7900 -7894 -7885 -7858 -7789 -7760 -7731 -7710 -7708 -7674 -7671 -7658 -7643 -7601 -7597 -7582 -7555 -7555 -7516 -7450 -7433 -7432 -7390 -7363 -73
61 -7244 -7196 -7192 -7112 -7108 -7029 -6963 -6961 -6922 -6912 -6817 -6772 -6655 -6574 -6528 -6498 -6482 -6362 -6354 -6319 -6270 -6256 -6230 -6195 -6194 -616
3 -6120 -6028 -5980 -5909 -5854 -5677 -5544 -5482 -5324 -5197 -5151 -5138 -4989 -4755 -4742 -4701 -4666 -4597 -4574 -4568 -4539 -4519 -4483 -4482 -4472 -4413
-4298 -4176 -4081 -4077 -4025 -3976 -3947 -3942 -3929 -3842 -3831 -3798 -3792 -3756 -3689 -3670 -3643 -3639 -3600 -3490 -3483 -3418 -3357 -3282 -3278 -3277
-3200 -3129 -3119 -3012 -2906 -2880 -2843 -2842 -2825 -2820 -2804 -2802 -2802 -2794 -2780 -2704 -2670 -2624 -2465 -2398 -2297 -2294 -2270 -2267 -2266 -2245 -2
2162 -1968 -1933 -1905 -1852 -1843 -1812 -1790 -1777 -1760 -1713 -1709 -1646 -1644 -1615 -1551 -1491 -1408 -1396 -1389 -1369 -1335 -1330 -1322 -1321 -1274 -1
273 -1172 -1009 -993 -975 -905 -750 -738 -715 -666 -631 -542 -526 -470 -382 -371 -262 -250 -222 -215 -155 -106 -101 -40 -22 16 23 33 41 45 67 171 219 238 239
281 372 404 447 454 463 614 621 630 652 763 794 802 823 824 878 883 1029 1058 1063 1128 1141 1191 1211 1213 1304 1338 1370 1445 1446 1457 1496 1508 1509 162
5 1667 1715 1753 1754 1767 1831 1857 1893 1906 1912 1919 1943 1949 2019 2029 2045 2060 2192 2196 2251 2264 2278 2306 2338 2393 2394 2404 2477 2590 2635 2723
2765 2805 2816 2818 2892 2901 2904 2911 3031 3058 3073 3099 3103 3197 3265 3290 3291 3309 3376 3379 3426 3488 3512 3562 3594 3735 3783 3807 3809 3830 3858 38
69 3893 3964 3966 3977 4057 4078 4126 4138 4189 4243 4249 4252 4284 4305 4309 4355 4366 4376 4383 4413 4418 4462 4505 4513 4523 4727 4782 4782 4786 4847 4934
4949 4997 5034 5034 5065 5169 5201 5246 5247 5335 5337 5393 5405 5439 5465 5514 5532 5585 5668 5713 5733 5750 5769 5815 5909 5917 6008 6062 6078 6080 6232 6
238 6290 6299 6355 6362 6385 6427 6429 6527 6558 6588 6638 6652 6726 6774 6813 6831 6841 6865 6886 6890 6891 6896 6921 6978 7005 7037 7054 7072 7118 7146 722
0 7341 7350 7413 7436 7481 7519 7534 7610 7647 7708 7720 7724 7755 7818 7822 7843 7852 7864 8010 8026 8028 8085 8102 8108 8125 8153 8302 8331 8356 8361 8366
8426 8560 8572 8583 8586 8614 8666 8672 8731 8764 8836 8885 8891 8938 8970 9029 9073 9076 9108 9135 9164 9264 9266 9283 9309 9310 9312 9321 9404 9447 9484 94
90 9510 9557 9590 9639 9645 9683 9695 9709 9714 9726 9771 9776 9782 9808 9810 9912 9964
spent time: 0 seconds
```

測資書：652 時間：0.001s

```
C:\Users\chuch\Desktop\HW2_109550027\Mergesort_109550027.exe
-5469 -5452 -5435 -5424 -5396 -5391 -5376 -5354 -5351 -5323 -5268 -5224 -5100 -5060 -5013 -4998 -4985 -4973 -4924 -4910
-4909 -4877 -4876 -4863 -4826 -4788 -4779 -4773 -4762 -4717 -4709 -4688 -4548 -4453 -4435 -4408 -4354 -4344 -4322
-4321 -4293 -4289 -4286 -4279 -4275 -4195 -4126 -3967 -3955 -3944 -3919 -3917 -3915 -3905 -3880 -3874 -3854 -3827 -3805
-3792 -3752 -3701 -3630 -3628 -3581 -3579 -3518 -3467 -3388 -3374 -3332 -3307 -3300 -3293 -3262 -3245 -3231 -3218 -3206
-3188 -3131 -3082 -3076 -3047 -3041 -3016 -2982 -2977 -2941 -2932 -2876 -2853 -2843 -2837 -2791 -2761 -2741 -2655 -2615
-2571 -2569 -2546 -2530 -2527 -2514 -2494 -2486 -2454 -2394 -2358 -2293 -2274 -2269 -2263 -2248 -2242 -2155 -2123
-2071 -2059 -2028 -1969 -1963 -1952 -1923 -1891 -1833 -1830 -1736 -1717 -1714 -1681 -1670 -1643 -1637 -1630 -1547 -1540
-1540 -1481 -1441 -1427 -1379 -1308 -1307 -1237 -1231 -1201 -1194 -1193 -1182 -1164 -1134 -1070 -1068 -1068 -1036 -1035
-1035 -1000 -964 -946 -946 -870 -853 -826 -773 -660 -655 -647 -622 -607 -570 -540 -529 -458 -451 -436 -421 -416 -345 -32
2 -292 -172 -121 -85 5 6 73 82 140 183 196 250 276 298 306 372 435 444 495 532 607 656 678 700 711 718 748 803 851 862 8
64 915 966 972 983 1001 1002 1012 1015 1040 1048 1077 1100 1232 1342 1382 1455 1515 1552 1553 1558 1613 1624 1814 1830 1
917 1924 1953 1966 2015 2018 2028 2040 2042 2111 2123 2126 2137 2164 2187 2197 2237 2306 2314 2320 2424 2441 2456 2459 2
460 2501 2568 2658 2659 2695 2712 2714 2748 2758 2772 2784 2786 2798 2829 2876 2897 2897 2957 2983 3093 3095 3171 3
220 3261 3265 3336 3341 3418 3422 3429 3452 3453 3468 3510 3522 3548 3578 3587 3615 3623 3691 3693 3778 3805 3867 3928 3
930 3997 4023 4139 4144 4184 4221 4396 4406 4427 4522 4540 4571 4584 4588 4598 4613 4619 4642 4685 4747 4776 4841 4841 4
862 4950 4995 5031 5050 5144 5155 5162 5199 5288 5301 5310 5315 5327 5422 5470 5532 5538 5581 5607 5698 5788 5813 5835 5
835 5888 5891 5946 5955 5957 6001 6045 6046 6091 6145 6147 6148 6152 6192 6199 6209 6221 6231 6234 6248 6335 6340 6368 6
399 6411 6417 6429 6490 6493 6511 6550 6560 6569 6583 6628 6652 6695 6705 6721 6739 6750 6753 6761 6792 6809 6810 6834 6
875 6900 6919 7033 7048 7060 7145 7157 7161 7192 7194 7255 7265 7275 7349 7353 7361 7382 7391 7396 7406 7424 7443 7447 7
482 7525 7536 7564 7588 7590 7592 7596 7706 7731 7732 7752 7766 7788 7850 7919 7926 7963 8041 8042 8076 8146 8160 8190 8
202 8209 8235 8246 8308 8310 8316 8381 8393 8505 8513 8515 8541 8579 8650 8651 8665 8732 8772 8804 8851 9025 9033 9041 9
043 9077 9081 9180 9188 9198 9205 9325 9330 9340 9358 9364 9365 9390 9412 9414 9416 9443 9499 9534 9622 9727 9744 9767 9
789 9807 9849 9857 9876 9899 9901 9940 9998
spent time: 0.001 seconds
0
-----
Process exited after 11.74 seconds with return value 0
請按任意鍵繼續 . . .
```

- Time complexity
 - Quicksort: $O(n \log n)$
 - Mergesort: $O(n \log n)$
- Discussion
- Your discover

Sorting 的部分好像蠻容易的，但是覺得他蠻厲害的畢竟用 bubble sort 什麼的很笨。然後計算 time 的部分真的快搞死我，他一開始時間差全部都是零，我自我懷疑超久，後來尋求助教協助，助教說是我的測資不夠大。之後跑很大的測資才開始有一些數字跑出來，可是我要測卻沒辦法一次貼上那麼多，例如測資二測資三，只能一次貼上一筆數據。

- Which is better algorithm in which condition

如果相同的測資比較多，或者是 reverse case，用 quick sort 比較快。空間不夠的話，用 quick sort 也比較省空間。

- Challenges you encountered

- Mergesort

```
for(k=1;k<r+1;k++){//&&i<(m-l+1)&&j<(r-m)
    if(i>=(m-l+1)){
        A[k] = right[j++];
        continue;
    }
    if(j>=(r-m)){
        A[k] = left[i++];
        continue;
    }
    if(left[i]<=right[j]){
        A[k] = left[i++];
    }
    else if(left[i]>right[j]){
        A[k] = right[j++];
    }
}
```

一開始沒加藍色框框，在其中一邊已經全部加進去，另一邊還沒加完的情況下，另一邊沒辦法加進去，結果出現很多 0。

- Conclusion
- What did you learn from this homework

o How many time did you spend on this homework

o Feedback to TA

大概都花了兩三個小時，學到最重要的就是這兩個演算法，對於降

低 time complexity 很有幫助。也學到了 time.h 的應用。

Part two: MST

- Introduction

- Introduction of different method/ algorithm

這部分實作了兩個 MST: Prim 和 Kruskal，兩者都 Greedy algorithm，

Prim 著重在 vertices 上；Kruskal 著重在 edges 上。

- Implement Details

- Steps

- Prim

創建了一個 class: graph，把 prim 的所有功能包在裡面。Graph

有兩個變數 v for vertex index, vector<intpair> array myadj。

myadj[]裡面用一個 intpair(int,int)存取 adjacent vertex 的 index

和自己與該 vertex 之間的 edge cost。

利用 addedge()將 endnode push_back 到 startnode 的 myadj，

同時也將 startnode push_back 到 endnode 的 myadj。

在 prim()裡面，先建立一個 minheap pririty queue，存取 intpair

(key, vertex_index)，key 存在前面是因為 priority queue 會依照

第一個 element 的大小去 sort queue。接著建立三個 vector:

<bool> isvisit, <int> parent, <int> key，並讓 key[0]=0, 其他=

infinity，之後 push 到 priority queue 內。

最後透過 while 迴圈以及迭代器，將 pop 掉的 unvisit vertex 的

adgacent vertex 進行更新 key 和 parent 的動作，更新依據：

key[鄰居]是否大於 edge weight，最後將更新後的 key push 回

priority queue，以便下回可以再將 weight 最小的 unvisit vertex pop 掉。

■ Kruskal

首先，我利用創了一個 `std::set<int>` 的 `array:sets[]`，每一個 vertex 都是一個 set。

再來，我創了一個 `class: edge`，裡面存放三個 `int` 變數：

`startnode, endnode, weight`，以及一個 `function: sort edge()`，

為了比較 `edge` 的大小，我 `overloading "<" operator`，告訴他

`edge` 的大小想要依照 `weight` 的大小做比較，並使用 `std::sort` 去處理。

接著，我寫了一個 `int function: find set()`。傳入的值有 `sets`

`array, sets array size`，和我要找的 vertex `v`。利用 `for` 迴圈以及

`find()`，若我要找的 `v` 在 `sets[i]` 裡面，就會傳回 `i` 的值。

接著，在 `main` 裡面用 `for` 迴圈讓每一個 `edge` 跑一次(`weight` 從

低至高的 `edge`)，讓每個 `edge` 的 `startnode` 及 `endnode` 做 `find`

`set`，若兩者 `sets[i]` 的 `i` 值不同，則進行 `union` 的動作。

`Union function`：若兩個 `set` 不相同，則將比較小的 `set` 併入比

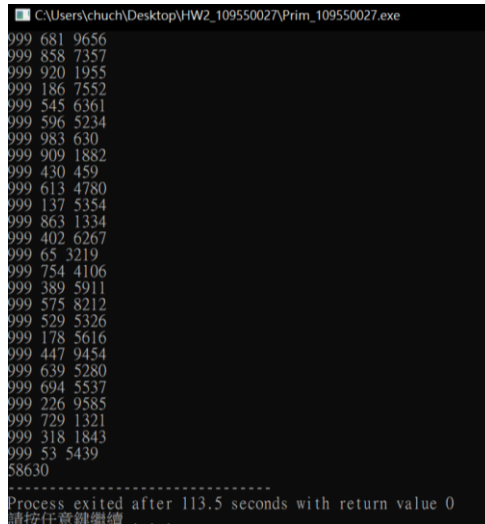
較大的 `set`。我利用 `set1.insert(set2.begin(),set2.end())` 將其合

併，還要記得 `set2.clear()` 才不會在下次 `find set` 出現問題。

- Results

- Prim(測資一) (vertex,edge)=(1000,99822)

結果：58630 Time complexity： $O(E \log V)$

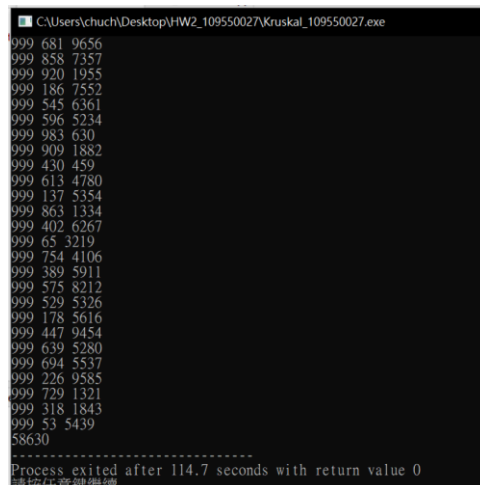


```
C:\Users\chuch\Desktop\HW2_109550027\Prim_109550027.exe
999 681 9656
999 858 7357
999 920 1955
999 186 7552
999 545 6361
999 596 5234
999 983 630
999 909 1882
999 430 459
999 613 4780
999 137 5354
999 863 1334
999 402 6267
999 65 3219
999 754 4106
999 389 5911
999 575 8212
999 529 5326
999 178 5616
999 447 9454
999 639 5280
999 694 5537
999 226 9585
999 729 1321
999 318 1843
999 53 5439
58630
-----
Process exited after 113.5 seconds with return value 0
請按任意鍵繼續
```

- Kruskal

(測資一) (vertex,edge)=(1000,99822)

結果：58630 Time complexity： $O(E \log E)$



```
C:\Users\chuch\Desktop\HW2_109550027\Kruskal_109550027.exe
999 681 9656
999 858 7357
999 920 1955
999 186 7552
999 545 6361
999 596 5234
999 983 630
999 909 1882
999 430 459
999 613 4780
999 137 5354
999 863 1334
999 402 6267
999 65 3219
999 754 4106
999 389 5911
999 575 8212
999 529 5326
999 178 5616
999 447 9454
999 639 5280
999 694 5537
999 226 9585
999 729 1321
999 318 1843
999 53 5439
58630
-----
Process exited after 114.7 seconds with return value 0
請按任意鍵繼續
```

- Discussion

- Your discover

最重要的發現是，很多東西我原本以為他只是一個名字，要自己去定義，結果查了 cplusplus.com 以後才發現他們都是內建的 function...

舉例 `priority_queue`、`set` 等等，她們就像好工具一樣會讓速度快很多，工欲善其事必先利其器，所以平常還是要多打扣來增廣見聞。再來就是我發現除了之前交的 `sorting` 之外，還有很多 `function` 都附帶有排序的功能，例如上面講的 `priority_queue` 和 `set`，除此之外還發現 `std::sort` 不一定只能按數字大小排，也可以透過 `function` 或者是 `object` 來排(第三個 `parameter`)，而我這次選擇了 `operator overloading`，就不需要第三個 `parameter`。

○ Which is better algorithm in which condition

當 $V \gg E$ 時，`kruskal` 的 Time complexity 較小，反之則 `Prim` 較小。

○ Challenges you encountered

■ `Prim`

原本我一直 `push` 新的 `key` 值到 `priority queue`，但舊的還沒有 `pop` 掉(也不知道怎麼 `pop` 因為他不是最小的值，例如 `infinity`)後來就發現可以先讓 `current v = pop` 掉的值，再接 `if current v is visit: continue`，這樣子那些 `inf` 或其他值最終還是會 `pop` 完！

■ `Kruskal`

在寫 `union()`時，一直無法對我的 `set` 做改變，後來索性把它丟到 `main` 裡面然後就好了，後來想想應該是沒有用好 `pointer` 的關係。還有一個讓我找超級久的 `bug`，就是在確認 `findset(start) != findset(end)`之後，一開始我讓 `a=edge.start`, `b=edge.end`，然後去比較還有 `union sets[a], sets[b]`，但後來我

發現 a 和 b 應該要是 find set 之後的值，而不是 edge 的值，真的好笨喔而且他好難找.....

- Conclusion

- What did you learn from this homework
- How many time did you spend on this homework
- Feedback to TA

- Prim

大概花五六個小時左右，學到了什麼是 pair, make pair，還有怎麼樣定義整數的無限大(但不是真的無限大)，也學了 priority queue 和怎麼把它改成小到大(„greater)，除此之外，本來對 iterator 沒有很了解，也因為這次機會去了解了。

- Kruskal

花了七八個小時吧，雖然題目也沒有到很難主要是 debug 超久。學到的東西：sort, set 一些小東西(都打在 discover 了)。

結論：這兩題都打好久喔連 report 也是，但是看到那一大串測資在跑而且跑出對的就覺得好開心，但我還是很累。

Part three: shortest path

- Introduction

- Introduction of different method/ algorithm

這部分實作 single source shortest path，和 MST 的差別是不同的起點會造成不同結果。

- Implement Details

- Steps

- Dijkstra

因為 dijkstra 也是著重在 vertices 上，所以一樣使用 Class

Graph 以及兩個變數 v for vertex index, vector<intpair> array

myadj。myadj[]裡面用一個 intpair(int,int)存取 adjacent vertex

的 index 和自己與該 vertex 之間的 edge cost。

利用 addedge()將 endnode push_back 到 startnode 的 myadj，

因為是 directed weight 所以不必將 startnode push_back 到 endnode 裡。

在 dijkstra()裡面，先建立一個 minheap priority queue，存取

intpair (distant, vertex_index)。接著建立三個 vector: <bool>

isvisit, <int> parent, <int> distant，並讓 starting vertex 的

distant=0, 其他= infinity，之後 push 到 priority queue 內。

利用 while 迴圈，若 Q!=empty()，也就是說若還沒 visit 全部 vertex，則依序將 vertex pop()並且更新他的鄰居們。更新的依

據：如果 distant[current vertex]+edge weight<distant[鄰居]，就

讓鄰居的 $\text{distant} = \text{distant}[\text{current vertex}] + \text{edge weight}$ 。

最後將 current vertex 的 isvisit 改為 1。最後因為題目要求印出

s 到 t 的 shortest path，其實就是印出 t 的 distant 值。

■ BellmanFord

Bellmanford 是從 edge 的角度去更新資料，所以沿用 class:

edge，裡面存放三個 int 變數：startnode, endnode, weight。

在 $\text{bellmanoford}()$ 裡面，建立一個 vector distant 存取 vertex 的資料，讓 $\text{starting vertex} = 0$, 其他 = infinity。

接著進行 $V-1$ 次的資料更新。對所有 edge，檢查 $\text{distant}[\text{start}]$

+ weight 是否小於 $\text{distant}[\text{end}]$ ，若較小，則使 $\text{distant}[\text{end}] =$

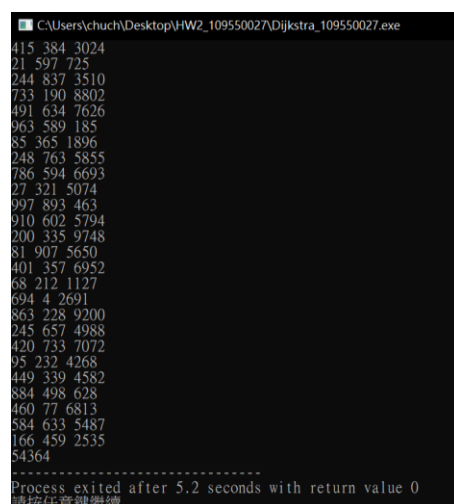
$\text{distant}[\text{start}] + \text{weight}$ 。(negative-weight cycle detect 在

discussion)。最後，只要印出 t 的 distant 值就好了。

● Results

■ Dijkstra(測資二) (vertex,edge)=(1000,2000)

結果：54364 Time complexity： $O(V^2)$



```
C:\Users\chuch\Desktop\HW2_109550027\Dijkstra_109550027.exe
415 384 3024
21 597 725
244 837 3510
733 190 8802
491 634 7626
963 589 185
85 365 1896
248 763 5855
786 594 6693
27 321 5074
997 893 463
910 602 5794
200 335 9748
81 907 5650
401 357 6952
68 212 1127
694 4 2691
863 228 9200
245 657 4988
420 733 7072
95 232 4268
449 339 4582
884 498 628
460 77 6813
584 633 5487
166 459 2535
54364
-----
Process exited after 5.2 seconds with return value 0
請按任意鍵繼續 . . .
```

■ BellmanFord

(測資二) (vertex,edge)=(1000,2513)

結果：24585 Time complexity：O(VE)

```
C:\Users\chuch\Desktop\HW2_109550027\BellmanFord_109550027.exe
588 223 2023
411 323 9714
13 338 5960
858 368 -67
381 811 7641
711 121 6223
41 843 7744
623 733 4485
770 998 8722
113 581 8477
396 405 1045
918 658 5091
127 722 9662
168 108 5235
690 578 -563
403 637 1065
362 395 3458
564 494 8599
512 294 9295
358 396 9448
949 667 5937
579 135 8605
880 419 8243
477 736 7652
315 287 5625
126 167 3228
24585
-----
Process exited after 6.363 seconds with return value 0
請按任意鍵繼續 . . .
```

(測資三) (vertex,edge)=(1000,3000)

結果：Negative loop detected! Time complexity：O(VE)

```
C:\Users\chuch\Desktop\HW2_109550027\BellmanFord_109550027.exe
372 266 1289
629 96 2266
11 622 -1746
986 677 1777
347 130 2006
61 330 6029
917 784 5505
744 989 2064
437 153 7096
775 604 3564
190 54 9952
138 143 4743
52 635 3278
926 950 2850
253 884 1939
188 804 5766
166 263 4408
791 856 8268
488 649 1111
706 943 9527
74 605 565
276 708 2560
99 42 3471
60 418 2686
248 566 9396
88 152 732
Negative loop detected!
-----
Process exited after 8.323 seconds with return value 0
請按任意鍵繼續 . . .
```

● Discussion

○ how to print out the path of the shortest path

這是一個很好的問題，因為我本來有寫一個 vector 去存 parent，可是後來發現都沒有用到就把他刪了。回到問題，如果我要印出整個 path 的話，我必須要知道她的 parent 是誰，然後從兒子一直往

parent 推，直到推回 starting vertex。

o Your discover

■ BellmanFord

● How to detect negative loop

結束 $V-1$ 次的檢查後，最後再進行第 V 次的檢查，正常情況下

$V-1$ 次就可以產生 shortest path，如果第 V 次數據還會改變的

話，也就是 $\text{distant}[\text{start}] + \text{weight} < \text{distant}[\text{end}]$ ，就代表有

negative-weight cycle，才會讓他每跑一次都比本來值的小。

o Time complexity

■ Dijkstra: $O(V^2)$

■ BellmanFord: $O(VE)$

o Which is better algorithm in which condition

如果 V 比 E 大很多的話，用 bellman ford 會比較快；此外，如果有

negative weight cycle 的可能的話，一定不能選擇 dijkstra。(bellman

ford 也只能偵測它的存在而已，不能計算。)

o Challenges you encountered

■ BellmanFord：一開始 if 裡面沒有寫 if $\text{distant}[\text{start}] \neq \text{inf}$ ，然後

結果一直不對，把過程印出來以後發現他的數字都沒改，猜想

可能是在最大 int_max 附近做加減導致問題。還有本來是照著

講義，negative weight cycle 的判斷式 if $\text{distant}(\text{end}) + \text{weight} \neq$

$\text{distant}(\text{start})$ ，結果怎麼跑結果都是 negative weight cycle，後

來想到應該要改成小於才對，因為本來就不會每個點都是

shortest path 更不可能永遠左右相等。

- Conclusion

- What did you learn from this homework
- How many time did you spend on this homework
- Feedback to TA

心得：寫的時候，Dijkstra 是參考 prim，因為兩者都是用 vertex 做比較；而 bellman 是參考 kruskal，他們都是讓 edge 跑迴圈，比較 start node 及 end node。因為先寫了 MST，讓這兩題變得容易許多，大概都花一個小時就完成了，其中最困難的應該就是怎麼拼出 Dijkstra 了吧！