General Function : swap-anything (any data type)
↳ swap them.

Generics → Java
Templates → C++ (User Defined Types)
(Intellisense)

int arr[] = [12, 19, 16, 49, 32]
0 1 2 3 4
100 104 108 112 116
↓ 4 bytes
↓ contiguous
arr → 100

function( ) {
a = 20
↓ 2000 &a →
}

main( )
int * ptr = function();
&a = ptr
*ptr = 2000;

Storage Classes :→
(i) static → global (file)
(ii) extern → global (folder)
(iii) auto → local
(iv) register → RAM → Cannot use pointers.
→ iterator (C++)
Iterator → Interface (Java)

folder: f1.c (static a, extern b) → f2.c (✗, ✓)

arr = {5, -8, 4, 1, -1, 2};
0 1 2 3 4 5
cmax = arr[0] = 5 ~~3~~ 4 5 ④ 6
gmax = arr[0] = 5 ⑥

Kadane's Algorithm
Maximum Subarray Sum
O(n) → Linear
arr[i], cmax + arr[i]
-8 , -3
4 , 1
1 , 5
-1, 4
2, 6

```
for (int i = 1; i < size; i++){
    cmax = max (arr[i], cmax + arr[i]);
    gmax = max (cmax, gmax);
}
return gmax;
```

DRY RUN

* Count Sort Algorithm :→
→ Non-comparison algo
→ Single digit whole nos. (0 – 9)
S1 :→ Find the max = 6
S2 :→ Create a count array 0 – max (0 – 6)
S3 :→ Calculate the freq of each element
S4 :→ Cumulative freq c = c + p
S5 :→ O/P array
S6 :→ Start from R → L

i/p: | 3 | 4 | 6 | 1 | 4 | 3 | 2 | 3 | 1 |
count: | 0 | 2 | 1 | 3 | 2 | 0 | 1 | → max
0 1 2 3 4 5 6
L→R Cumulative count: | 0 | 2 | 3 | 6 | 8 | 8 | 9 |
0 1 2 3 4 5 6
| 1 | 1 | 2 | 3 | 3 | 3 | 4 | 4 | 6 |
0 1 2 3 4 5 6 7 8
O(n + max)

Radix Sort Algorithm :→ (Bucket Sort) 1's, 10's, 100's ..
* Works for multi-digits
* Constant length strings [Rahul, Sunil, Samar]
Step 1 :→ Find the max
432 → digits → ③
3 iterations
Step 2 :→ 0 – 9 [10 buckets]
O(n + max)
P_1, P_2, P_3
CountSort (arr, size, exp)

| 325 | 432 | 021 | 009 | 087 |
P1 | 021 | 432 | 325 | 087 | 009 |
P2 | 009 | 021 | 325 | 432 | 087 |
P3 | 009 | 021 | 087 | 325 | 432 |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
P1 → 021 432 325 087 009
P2 → 009 021 432 325 087
P3 → 009 021 087 325 432
(12345)

* How do you control the no. of iterations in the "Radix Sort Algo" ?
max = 432 → ③
exp = 1
(1, 2, 3) → stop

```
for (int exp = 1; max / exp > 0; exp *= 10)
{
    countSort (arr, n, exp);
}
```

m = 432
e = 1, 10, 100, 1000
432/1 = 432
432/10 = 43
432/100 = 4
432/1000 = 0

Bubble Sort :→
| 7 | 6 | 4 | 3 |
① 6 7 4 3
6 4 7 3
6 4 3 7 → nth
0 1 n-2 n-1
①
j = 0, j < n-1
(second last)

② 6 4 3 | 7
4 6 3 7
4 3 6 7
n-1, (nth) OOB (last) ②

(4) → 3 iterations
(n-1) Iterations
4 3 | 6 7
3 4 6 7
↓ sorted
f → it ③ O(n)
f → com O(n)
O(n^2)

Selection Sort :→
minIndex = i
j = i+1 to < n
minIndex = j
if (minIndex != i)
swap (arr[i], arr[minIndex])

⑤ 3, 2, 1, 4, 0
i = 0 1 2 3 4 5
0, 3, 2, 1, 4, 5
i j
O(n^2)
iterations
com/swap

Insertion Sort :→ ②
arr = [3, 2, 6, 8, 1, 5, 4]
0 1 2 3 4 5 6
8 [2] 2 3 6 8 1 5 4
temp = arr[i]; ✗ [1] 2 3 5 6 8
if (arr[j] > temp)
arr [j+1] = arr[j]
O(n^2)

i = 1 to i < n
j = (i - 1)
= 0
j-- = 0 - 1 = -1
while (j >= 0 && arr[j] > temp)
j == 0

* Square root of a number using binary search. (Integral Part)

```
int squareRoot (int n) {
}
```

√n → 0 — n
0 — 36
s e
ans = -1
n = 36 → 6
n = 37 → 6
n = 65 → 8
m = (s+e)/2 (log N)
= 18
18×18 = 324 > n
e = m-1 0 — 17
17/2 = 8
8×8 = 64 > n 0 — 7
e = m-1 7/2 = 3
3×3 = 9 (ans = mid (3))
s = m+1 4 — 7
3+1 = 4 11/2 = 5
5×5 = 25 < n ans = mid(5)
s = m+1 = 5+6 = 6 6 — 7
6×6 = 36 return mid;

← N →
N/2 N/2
N/4 N/4 N/4 N/4
N/8
N/16
$N/2^0$, $N/2^1$, $N/2^2$
$N/2^k$ k = 0, 1, 2,
$k = \log_2 N$
(6+7)/2 = 13/2 = 6

(s+e)/2 integer
largest value :→ $2^{31} - 1$ (INT_MAX)
Smallest value :→ -2^{31} (INT_MIN)
(-ve) ✗
OOB
TLE
arr → INT_MAX index + INT_MAX index
mid → max + max — OOB (-ve)
What is the optimal formula for mid?
⇒ m = [s + (e-s)/2] = (2s + e - s)/2 = (s+e)/2

(Merge / Quick / Wave / Heap) Sort
↳ Heaps

Feedback :→ session code : 11942
bizoftictraining.com