$O(n^2)$

```
find closest pair of cities
```

$O(1)$

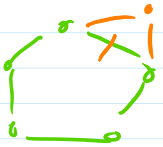
```
start partial tour with that closest pair

for i = 2 to n-1
    for each city from already in tour
        for each other city to
            compute distance, tracking closest/farthest

    city to insert is city to that gave closest/farthest distance

    for each insertion point j = 1 to i
        insert city to insert at position j in tour
        calculate distance of that tour, tracking minimum
    insertion point is j that gave minimum distance

    create new partial tour with city to insert inserted at position j
```



$O(n \lg n)$

worst case over all iterations
for $i = \frac{n}{2} - 1$

$\frac{n}{2} - 1$ outer loop
$\frac{n}{2} - 1$ inner loop
$O(n^2)$ total

$O(n)$ iterations

$O(n^2)$ per iter

$O(n)$ iterations

$O(n)$

$O(n^2)$ total

(can be $O(n)$ with both

$O(n)$

[can do in $O(1)$ w/ swaps from prev insert point]

[can do in $O(1)$ by updating dist of prev tour
− broken leg
+ 2 added legs]

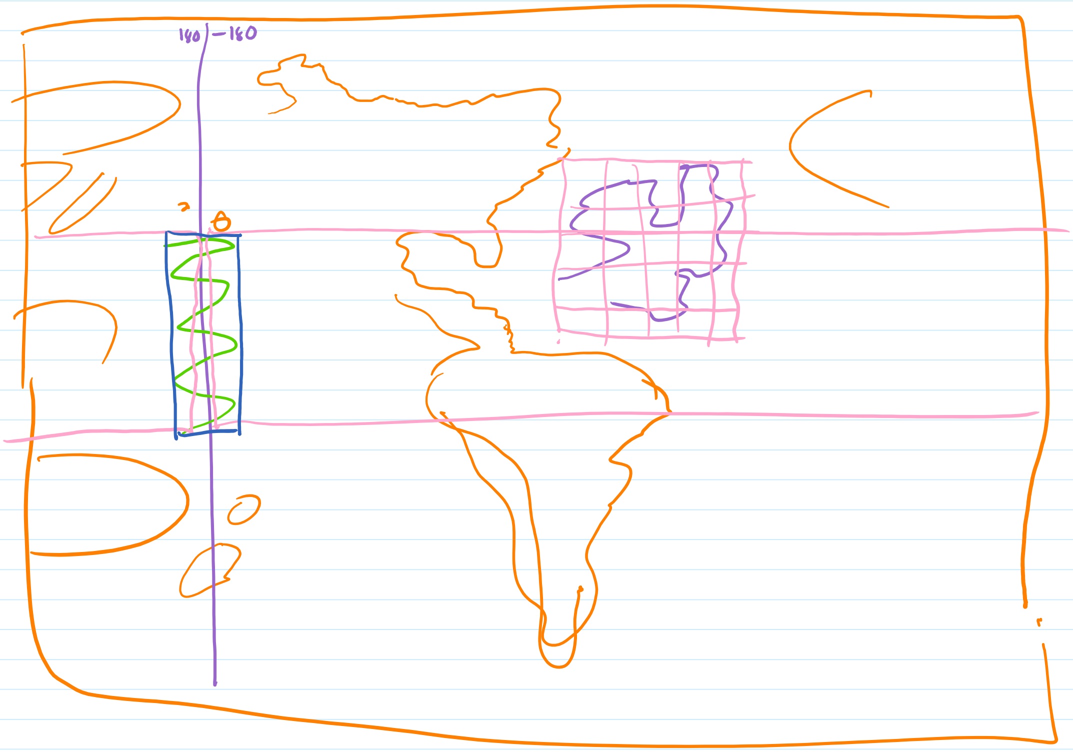
$O(n)$

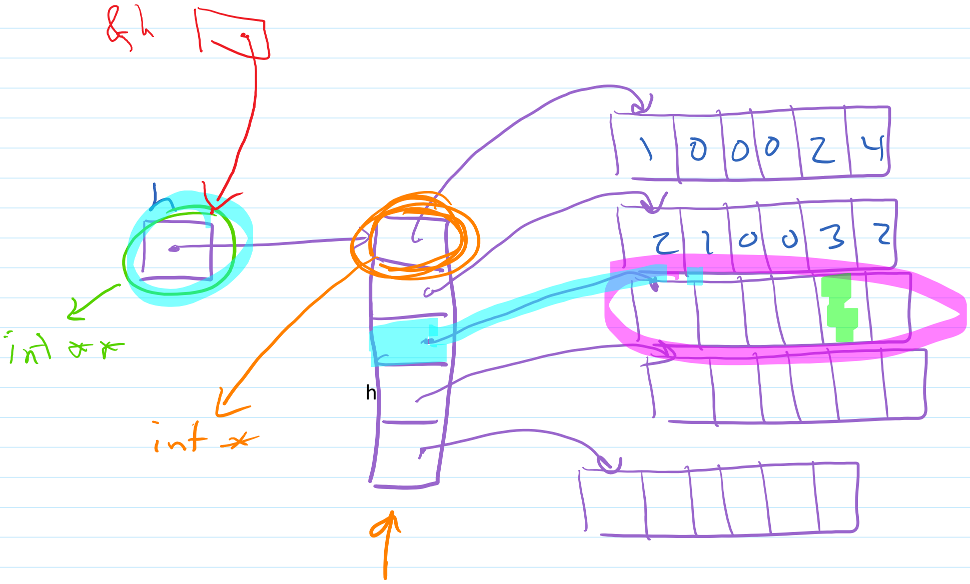
$$O(n^2 + 1 + n^3) = O(n^3) \text{ overall}$$

$O(n)$ iterations of for i loop
$O(n \log n + n) = O(n \log n)$ per iteration

$O(n^2 \log n)$ overall

Tuesday, October 8, 2019 3:18 PM



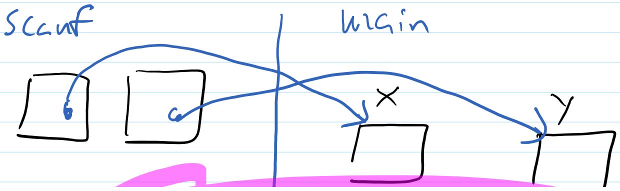


r=2
c=4

```
int **h = malloc(sizeof(int *) * rows);
for (int r = 0; r < rows; r++)
{
    h[r] = malloc(sizeof(int) * 6);
    for (int c = 0; c < cols; c++)
    {
        h[2][4] = 0;
    }
}
```

```
for (int r = 0; r < rows; r++)
{
    free(h[r]);
}
free(h);
```

```
double x;
double y;
scanf("%lf %lf", &x, &y);
```



```
double x;
double y;
scanf("%lf %lf", &x, &y);
```