Runtime of the G-S Algorithm:

- $O(n^2)$. Recall: we focus on *worst case* runtime.

  Because # of iterations $= n^2$

  All operations take $O(1)$ in our implementation.

- $\Theta(n^2)$ ?

  Yes! But we did not cover it in class.

  We did not prove runtime $= \Omega(n^2)$.

  To prove this, we need to construct inputs with n men and n women, such that our implementation of G-S algorithm takes $\Omega(n^2)$ time.

  - Input size $= \Theta(n^2)$

    Reading the input takes $\Theta(n^2)$.

  - Even in the model where we can ask "Who is $m_i$'s $j$-th favorite woman?" in $O(1)$ time., G-S algorithm may still take $\Omega(n^2)$ iterations (see Q1 in HW1).

---

Common runtimes:

Q1: Find maximum element in an array.

A1:
```
max = -∞
for i = 1 to n
    if A[i] > max then max = A[i];
```
runtime $= O(n)$

Q2: Decide if there is a consecutive interval that sums to s in the input array.

Example: $A = [1, -6, 2]$

$s = 3$ — No!

$s = -3$ — Yes! $\text{sum}[1, -6, 2] = -3$.

A2:
```
for i = 1 to n
    for j = i to n.
        sum_i_j = 0.
        for k = i to j.
            sum_i_j = sum_i_j + A[k].
        If sum_i_j = s return YES.
return No.
```
$O(1)$.

$O(n^3)$: 3 loops, each loop runs $O(n)$ time.

$\Omega(n^3)$: Yes! $\left(\frac{1}{3}n\right)^3$ $\quad i = 1 \dots \frac{n}{3}$, $\quad j = \frac{2n}{3} \dots n$.

$k = i \dots j$, there are $\geq \frac{n}{3}$ choices for $k$.

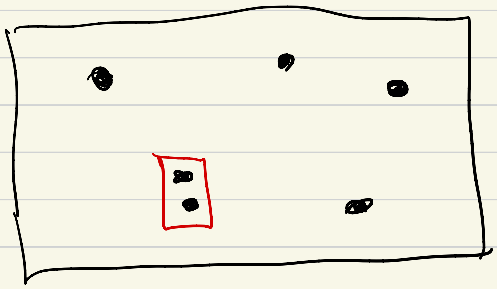
Q3: Find a consecutive interval with maximum average.

A3: Same as A2, except that we compute
$avg\_i\_j = sum\_i\_j / (j-i+1)$;
and output the $(i,j)$ with the maximum $avg\_i\_j$.

Runtime of A3 = $O(n^3)$.

This can be solve in $O(n)$ time.
Observe that maximum-average interval is always a single element.

Q4: Given $n$ points on 2-D plane,
find two points that are closest to each other.



A4:
```
min_dis = +∞
For i = 1 to n
    for j = 1 to n
        if (i ≠ j)
            min_dis = min(min_dis, sqrt((x_i - x_j)^2 + (y_i - y_j)^2))
```

```
min(a,b):
if a<b
    return a;
else
    return b;
```

Runtime of A4: $O(n^2)$.

This can be done in $O(n \log n)$
using a divide-and-conquer algorithm.