- Recursion
  \[ T(n) = 3T\left(\frac{n}{2}\right) + n \]

- Recursion tree method
  \[ \sum_{i=0}^{\log_2 n} \text{total merge cost at } i\text{-th layer} \]
  
  Merge cost for layer \( i \):
  \[ 4n \times 3^{\log_2 n} = 2^i n \]

- Often:
  \[
  \begin{cases}
  C < 1 & f(n) \text{ (no matter how many layers)} \\
  C = 1 & f(n) \cdot \#\text{layers (rough estimation on \# layers)} \\
  C > 1 & \#\text{layers} \cdot f(n) \text{ (accurate estimation on \# layers)}
  \end{cases}
  \]

  Answer is between \( n^{\log_2 3} \) and \( n^2 \)

  \[ T(n) = n^a \]
  \[
  n^a = 3\left(\frac{n}{2}\right)^a + 2\left(\frac{n}{4}\right)^a + n
  \]
  lower order term

  \[ 1 = 3\left(\frac{1}{2}\right)^a + 2\left(\frac{1}{4}\right)^a \]

  Solve for \( a \), and \( T(n) = \Theta(n^a) \) is the right solution.

(b)

\[
\begin{aligned}
T(n) &= F(\log_2 n) \\
F(k) &= 2T(2^{k-1}) + 2^k
\end{aligned}
\]

\[
T(n) = F(\log_2 n) = 2T(2^{\log_2 n - 1}) + 2^{\log_2 n}
\]

\[
= 2T\left(\frac{n}{2}\right) + n
\]

\[ T(n) = \Theta(n^{\log_2 3}) \]

\[ T(n, m) = 4T\left(\frac{n}{2}, \frac{m}{2}\right) + nm \]

Let \( k = nm \)
Let \( k = \frac{n}{2} \)
\[
T(k) = 4T\left(\frac{k}{4}\right) + k
\]

- Dynamic programming

State: \( a[i,j] \) = max number of pancakes we can get at time \( i \), character is at location \( j \) at time \( i \)

\[
a[i,j] = \max \begin{cases} a[i-1,j] & \text{stay} \\ a[i-1,j-1] & \text{move to the right (if } j > 1) \\ a[i-1,j+1] & \text{move to the left (if } j < n) \\ + 1 & \text{(if there is a pancake at location } j \text{ at time } i) \\ \end{cases}
\]

Output: \( \max_{j=1,2,...,n} a[m,j] \)

- Greedy

Idea 1: Consider rooms in increasing capacity, assign any group to the current room if possible.

Rooms: \( 3, 5, 7 \)
Groups: \( 4, 6 \)

Proof of correctness:

Wlog assume rooms and groups are in increasing order.

Assume towards contradiction that there is another solution with more meeting rooms assigned.

Let \( X_1, X_2, \ldots, X_k \) be the indices of rooms that are assigned in the algorithm.

Let \( Y_1, Y_2, \ldots, Y_{k'} \) be that are assigned in \( \text{OPT} \)

Because \( \text{OPT} \) is better, \( l' > l \)
assigned in OPT.

because OPT is better, $l' > l$

look at the first location where $X_i \neq Y_i$.

if $X_i > Y_i$,

in this case $Y_i$ is considered
before $X_i$ in the alg, and
the alg could not find any groups for $y_i$.

this should not be possible because there are at least
$i$ groups that meet the capacity limit of $Y_i$.

alg has only assigned $i-1$ of them, so there must be
one left.

else if $X_i \leq Y_i$,

want to argue: can change
OPT to agree with alg's assigned $Y_i$'s.

can fix $Y_i \rightarrow X_i$, and assign the same groups
to $Y_1, Y_2, \ldots, Y_i$.

this may conflict with some later assignments, but
these conflicts can be resolved because later rooms are bigger.

A more complete proof:

Proof: assume towards contradiction that OPT could schedule more meetings.

assume without loss of generality that rooms are sorted in increasing capacity.

let $X_i (i=1,2,\ldots,n)$ be the group assigned to meeting room $i$ for ALG.

let $Y_i (i=1,2,\ldots,n)$ be the group assigned to meeting room $i$ for OPT.

if room $i$ is empty in ALG/OPT, then $X_i = \emptyset$, $Y_i = \emptyset$.

let $i$ be the first index where $X_i \neq Y_i$.

if $X_i = \emptyset$ (ALG did not assign a group to room $i$, OPT did)

this is not possible because OPT assigned group $Y_i$ to room $i$, group $Y_i$ must still be available for ALG (because $X_j = Y_j$ for $j < i$), so the ALG should
\[ \text{this is not possible because } u(i) \text{ assigned group } y_i \text{ to room } i, \text{ given } y_i. \]

\[ \text{must still be available for ALG (because } x_j = y_i \text{ for } j < i \text{), so the ALG should also be able to assign a group to room } i. \]

\[ \text{else if } y_i = \emptyset \text{ (OPT did not assign a group to room } i\text{, ALG did)} \]

\[ \text{if the group } x_i \text{ was not assigned to any room,} \]

\[ \text{then we can additionally assign } x_i \text{ to room } i, \text{ resulting in a solution that's even better than OPT. Impossible.} \]

\[ \text{else, } x_i \text{ was assigned to room } j \text{ (} x_i = y_j \text{)} \]

\[ \text{can set } y_i = x_i, \ y_j = \emptyset, \text{ OPT remains valid (and is now closer to ALG)} \]

\[ \text{else (both ALG and OPT assigned a group, but the groups assigned are different)} \]

\[ \text{if } x_i \text{ was not assigned to any other room in OPT} \]

\[ \text{then we can just set } y_i = x_i \]

\[ \text{else: } x_i = y_j \text{ for some } j \]

\[ \text{then we can swap } y_i, y_j \text{, and still have } y_i = x_i. \]

\[ \text{repeat this argument, at the end OPT is the same as ALG, so OPT cannot be better than ALG.} \]