Alison Goshulak
CSC 225
Assignment 2

1. Step 1: $T(n) = 2T(\frac{n}{3}) + n$

Step 2: $T(n) = 2[2T(\frac{n}{9}) + \frac{n}{3}] + n$
$= 4T(\frac{n}{9}) + \frac{2n}{3} + n$

Step 3: $T(n) = 4[2T(\frac{n}{27}) + \frac{n}{9}] + \frac{2n}{3} + n$
$= 8T(\frac{n}{27}) + \frac{4n}{9} + \frac{2n}{3} + n$

Step i: $T(n) = 2^i T(\frac{n}{3^i}) + n \sum_{j=0}^{i-1} (\frac{2}{3})^j$

$= 2^i T(\frac{n}{3^i}) + n\left[\frac{1-(\frac{2}{3})^i}{1-\frac{2}{3}}\right]$

$= 2^i T(\frac{n}{3^i}) + 3n[1-(\frac{2}{3})^i]$ (1)

Set $\frac{n}{3^i} = 1 \Rightarrow n = 3^i \Rightarrow \log_3 n = i$

Substitute $i = \log_3 n$ into (1):

$T(n) = 2^{\log_3 n} T(\frac{n}{3^{\log_3 n}}) + 3n[1-(\frac{2}{3})^{\log_3 n}]$

$= 2^{\log_3 n} T(1) + 3n - 3n\left(\frac{2^{\log_3 n}}{n}\right)$

$= 3n - 2(2^{\log_3 n}) = 3n - 2^{\log_3 n + 1}$

2. Step 1: $T(n) = T(n-1) + \log_2 n$

Step 2: $T(n) = [T(n-2) + \log_2 (n-1)] + \log_2 n$

Step 3: $T(n) = [T(n-3) + \log_2 (n-2)] + \log_2 (n-1) + \log_2 n$

Step i: $T(n) = T(n-i) + \sum_{j=0}^{i-1} \log_2 (n-j)$

$= T(n-i) + \log_2 [n(n-1)\dots(n-(i-1))]$

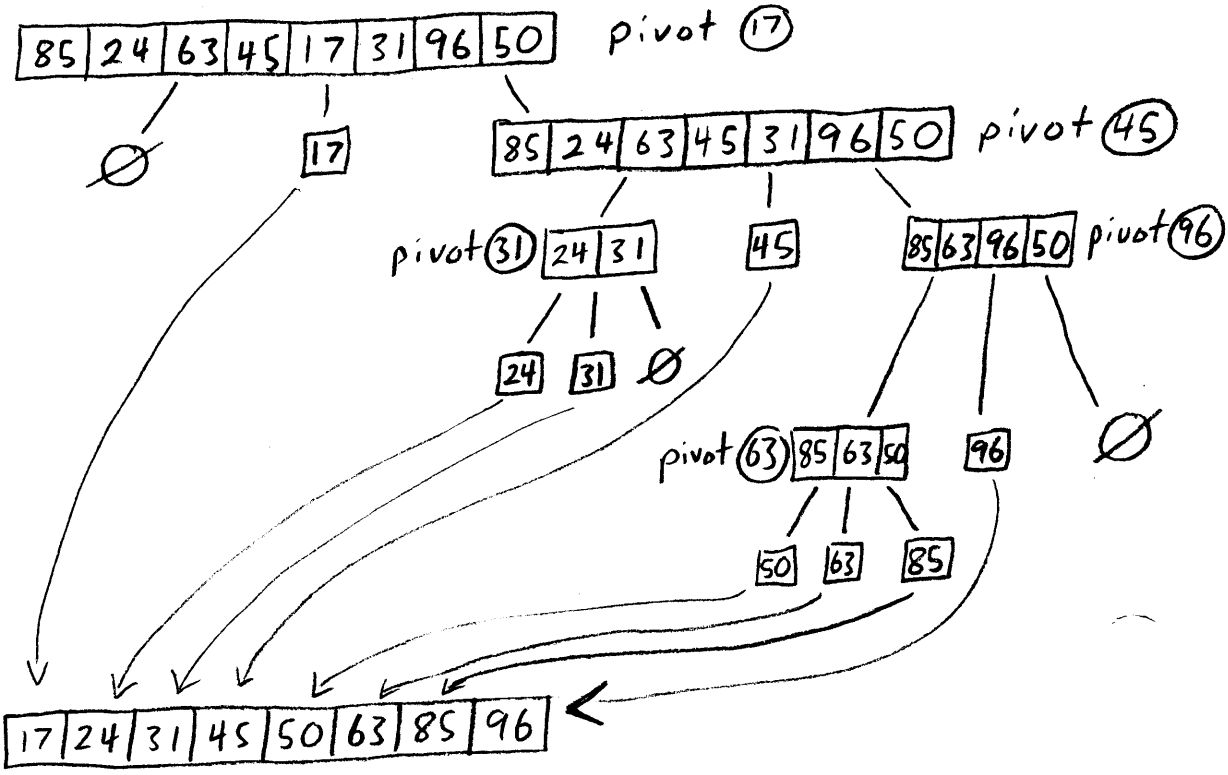
$= T(n-i) + \log_2 \frac{n!}{(n-i)!}$ ①

Set $n - i = 1 \Rightarrow i = n-1$

Substitute $i = n-1$ into ①:

$T(n) = T(n-(n-1)) + \log_2 \frac{n!}{(n-(n-1))!}$

$= T(1) + \log_2 \frac{n!}{1!} = 1 + \log_2 n!$

3. $S =$ 

4. Algorithm CountInversions(A)
Input: An array A
Output: The number of inversions in A

```
mid ← (length of A)/2
for i←0 to mid-1 do
    S1[i] ← A[i]
    i←i+1
end
index←0
for i← mid to (length of A)-1 do
    S2[index]←A[i]
    i←i+1
    index←index+1
end
inv←0
if (length of S1)>1 do
    inv← CountInversions(S1)
end
if (length of S2)>1 do
    inv←inv + CountInversions(S2)
end
return inv + MergeAndCount(A, S1, S2, 0, 0, 0)
```

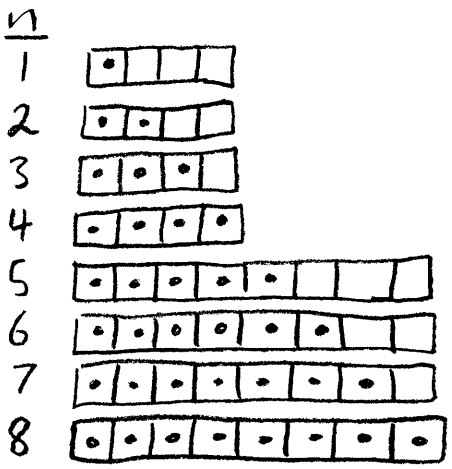
(MergeAndCount on next page ...)

(4) Algorithm MergeAndCount(A, S1, S2, i, j, k)
Input: Three arrays (A, S1, S2) and three ints (i, j, k)
Output: The number of inversions that cross between S1 and S2 (A also becomes sorted in the process)

```
inv ← 0
if j ≤ (length of S1)-1 AND k ≤ (length of S2)-1 do
    if S1[j] > S2[k] do
        inv ← (length of S1)-j
        A[i] ← S2[k]
        i ← i+1
        k ← k+1
    end
    else do
        A[i] ← S1[j]
        i ← i+1
        j ← j+1
    end
    inv ← inv + MergeAndCount(A, S1, S2, i, j, k)
end
else if k ≤ (length of S2)-1 do
    A[i] ← S2[k]
    i ← i+1
    k ← k+1
    inv ← inv + MergeAndCount(A, S1, S2, i, j, k)
end
else if j ≤ (length of S1)-1 do
    A[i] ← S1[j]
    i ← i+1
    j ← j+1
    inv ← inv + MergeAndCount(A, S1, S2, i, j, k)
end
return inv
```

5. Push cost for $f(N) = N + c$: (let $c = 4$)

with $n = 8$ push ops: (starting with $N = 0$)



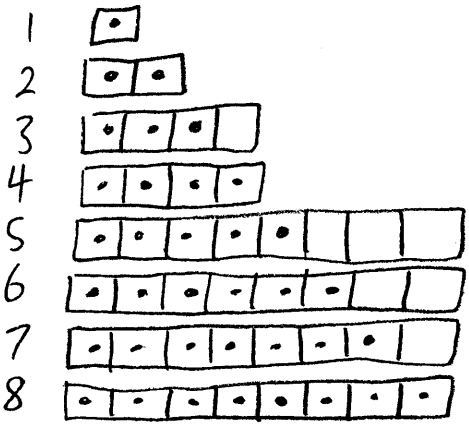

The cost of phase $k = 2 \cdot c \cdot k = 8k$ (where $k = \frac{n}{c} = \frac{n}{4}$)

Total cost of $K$ phases $= 8 \sum_{i=1}^{K} i = 8\left[\frac{(K+1)K}{2}\right] = 4K^2 + 4K$

$= O((\frac{n}{4})^2) = O(n^2)$

Push cost for $f(N) = 2N$:

with $n = 8$ push ops: (starting with $N = 1$)




The cost of phase $k = \begin{cases} 1 & \text{if } k = 1 \\ 2^k & \text{if } k \geq 2 \end{cases}$ (where $k = \log_2 n$)

Total cost of $K$ phases $= 1 + \sum_{i=2}^{K} 2^i = 1 + \left[\frac{2^{K+1} - 1}{2 - 1}\right]$

$= 2^{K+1}$

$= 2^{\log_2 n + 1} = 2n = O(n)$

Therefore, $f(N) = 2N$ is the better strategy for resizing the array