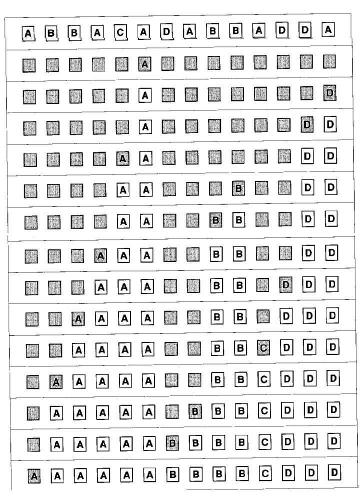
Linear Sorting Techniques

CLRS Reading: Sections 8.2, 8.3, 8.4, pages 168 -- 182
Problem Set: Assignment #5 due Friday, March 7

Sorting with Special Knowledge




```
Counting-Sort(X,Y,[A..D])
    for i ← A to D
        do count[i] ← 0
    for j ← 1 to length[X]
        do count[X[j]] ← count[X[j]] + 1
    for i ← 2 to k
        do count[i] ← count[i] + count[i-1]
    for j ← length[X] downto 1
        do Y[count[X[j]]] ← X[j]
           count[X[j]] ← count[X[j]] - 1
```

Fundamental Algorithms

Worst-Case Running Time of Counting-Sort

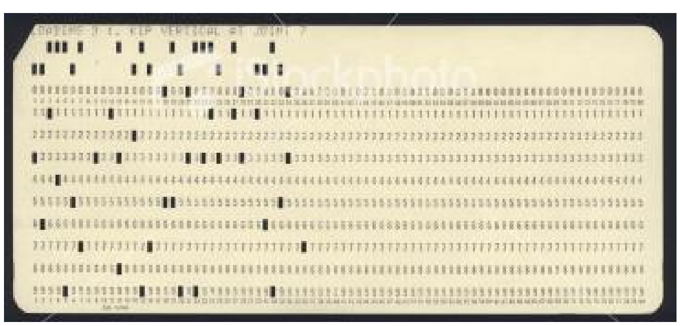
- Let *n* be the number of entries in *X* and let *k* be the number of possible keys in given range.
-
```
Counting-Sort(X,Y,[A..D])
    for i ← A to D
       do count[i] ← 0
    for j ← 1 to length[X]
       do count[X[j]] ← count[X[j]] + 1
    for i ← 2 to k
       do count[i] ← count[i] + count[i-1]
    for j ← length[X] downto 1
       do Y[count[X[j]]] ← X[j]
          count[X[j]] ← count[X[j]] - 1
```

J - 3

Fundamental Algorithms

Punch Card Sort

- There are two possibilities when sorting by column: most to least significant; or least to most significant digit. Suggestions?



J - 4

Seeing is Believing

```
Radix-Sort(A,d)
    for i ← 1 to d
        do use a stable sort
           to sort array A
           on digit i
```

672
291
306
244
839
021

input array	first digit	second digit	third digit

Worst-Case Running Time of Radix-Sort

Lemma 8.3.
Given *n* *d*-digit numbers in which each digit can take on up to *k* possible values, RADIX-SORT correctly sorts in $\Theta(d*(n+k))$ time.

```
Radix-Sort(A,d)
    for i ← 1 to d
        do use a stable sort
           to sort array A
           on digit i
```

Bucket Sort*

```
Bucket-Sort(A)
    n ← length[A]
    for i ← 1 to n
        do insert A[i] into list B[⌊n*A[i]⌋]
    for i ← 0 to n-1
        do sort list B[i] with insertion sort
    concatenate the lists B[0], B[1], ..., B[n-1]
```

	A
1	.38
2	.32
3	.63
4	.07
5	.30
6	.49

	B
0	
1	
2	
3	
4	
5	

*What might happen in the worst case?

Average Case Analysis of Bucket-Sort

	A
1	.38
2	.32
3	.63
4	.07
5	.30
6	.49

	B
0	
1	
2	
3	
4	
5	

Cost of inserting into buckets and concatenating results

Cost of insertion sort on i^{th} bucket given bucket contains n_i elements

$$T(n) = \Theta(n) + \Sigma_{0 \le i \le n-1} O(n_i^2)$$

Summed over all the buckets

Fundamental Algorithms

Average Case Analysis of Bucket-Sort

$$
\begin{aligned}
E[T(n)] &= E[\Theta(n) + \Sigma_{0 \le i \le n-1} O(n_i^2)] \\
&= E[\Theta(n)] + \Sigma_{0 \le i \le n-1} E[O(n_i^2)] \\
&= \Theta(n) + \Sigma_{0 \le i \le n-1} O(E[n_i^2]) \\
&= \Theta(n) + \Sigma_{0 \le i \le n-1} O(2 - 1/n)^* \\
&= \Theta(n) + n\,O(2 - 1/n) \\
&= ?
\end{aligned}
$$

*We are using the fact, established in the following slides, that $E[n_i^2] = 2 - 1/n$.

J - 9

Fundamental Algorithms

$n_i = \Sigma_{1 \le j \le n} X_{ij}$

where X_{ij} is the indicator function

$$
X_{ij} = \begin{cases} 1 & \text{if } A[j] \text{ falls into bucket } i \\ 0 & \text{otherwise} \end{cases}
$$

Some observations concerning X_{ij}: $E[X_{ij}] = 1/n$; X_{ij}: $E[X_{ij}^2] = 1/n$; and $E[X_{ij}X_{ik}] = E[X_{ij}]E[X_{ik}] = 1/n^2$.

J - 10

Since $n_i = \Sigma_{1\le j\le n} X_{ij}$

$$E[n_i^2] = E[(\Sigma_{1\le j\le n} X_{ij})^2]$$

$$= E[\Sigma_{1\le j\le n} \Sigma_{1\le k\le n} X_{ij} X_{ik}]$$

$$= E[\Sigma_{1\le j\le n} X_{ij}^2 + \Sigma_{1\le j\le n} \Sigma_{1\le k\le n, k\ne j} X_{ij} X_{ik}$$

$$= E[\Sigma_{1\le j\le n} X_{ij}^2] + \Sigma_{1\le j\le n} \Sigma_{1\le k\le n, k\ne j} E[X_{ij} X_{ik}]$$

$$= \Sigma_{1\le j\le n} 1/n + \Sigma_{1\le j\le n} \Sigma_{1\le k\le n, k\ne j} 1/n^2$$

$$= n (1/n) + n(n-1)\, 1/n^2$$

$$= 1 + (n-1)/n$$

$$= 2 - 1/n$$