CS261 Data Structures

Hash Tables

Hash-like Sorting

Hash Tables: Sorting

- Can create very fast sort programs using hash tables
- These sorts are not 'general purpose' but very efficient for certain situations
  - e.g. only works on positive integers in a particular range
- Examples:
  - Counting sort
  - Radix sort

Hash Table Sorting: Counting Sort

- Quickly sort positive integer values from a limited range
  - Count (tally) the occurrences of each value using HT
  - Recreate sorted values according to tally
- Example:
  - Sort 1,000 integer elements with values between 0 and 19
  - Count (tally) the occurrences of each value:

0 - 47	4 - 32	8 - 41	12 - 43	16 - 12
1 - 92	5 - 114	9 - 3	13 - 17	17 - 15
2 - 12	6 - 16	10 - 36	14 - 132	18 - 63
3 - 14	7 - 37	11 - 92	15 - 93	19 - 89

  - Recreate sorted values according to tally:

47 zeros, 92 ones, 12 twos, …

Counting Sort: Implementation

```
/* Sort an array of integers, each element no larger than max. */
void countSort(int * data, int n, int max) {
  int  i, j, k;
  /* Array of all possible values. – it is the hash table */
  int *cnt = malloc((max + 1) * sizeof(int));

  for( k-0; k < max; k++) cnt[i] = 0; /* initialize */

  for (i = 0; i < n; i++)       /* Count the occurrences */
    cnt[data[i]]++;             /*    of each value.     */

  /* Cnt holds the number of occurrences of numbers from 0 to max. */

  i = 0;                        /* Now put values     */
  for (j = 0; j <= max; j++)    /*   back into the array. */
    for (k = cnt[j]; k > 0; k--) data[i++] = j;
}

/* Integer itself is the hash index */
```

which bucket

i is for total array count (1000)

J is for range (0-19)

k is for count of each bucket

What's the complexity of this sort?

Radix Sort

- Has historical ties to ***punch cards***

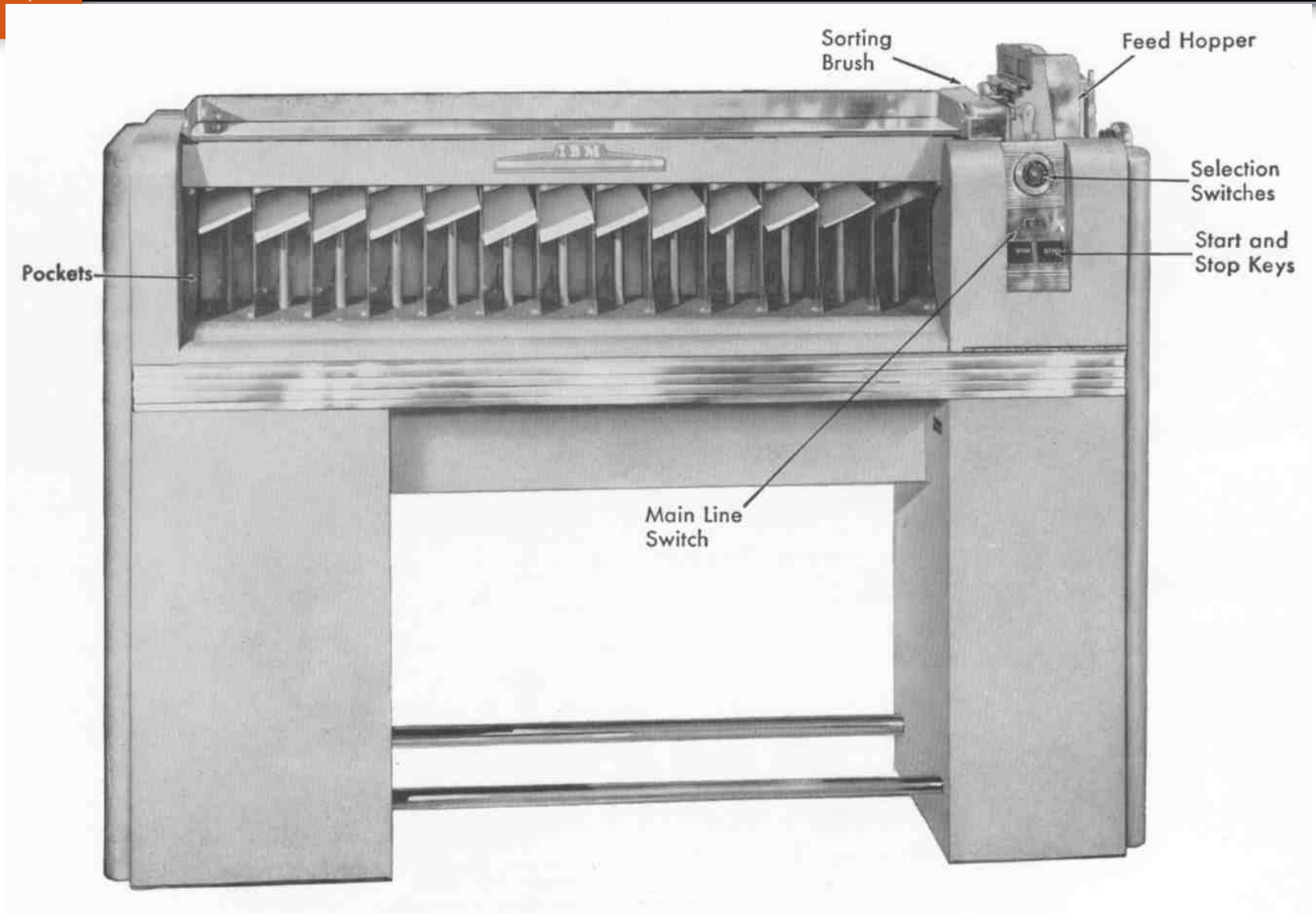
PunchCards

- Input for early digital machines
  - Data
  - Code
- Typically 80 columns of information
- Imagine writing code in the 60's

Sorting Punch Cards

- It was far too easy to drop a tray of cards, which could be a disaster
- Convention became to put a sequence number on card, typically in positions 72-80
- Could then be resequenced by sorting on these positions
- A machine called a sorter used to re-sort the cards

Mechanical Sorter: Sorts a Single Column



Mechanical Sorter

- First sort on column 80
- Then collect piles, ***keeping them in order,*** and sort on column 79
- Repeat for each of the columns down to 72
- At the end, the result is completely sorted
- Try it

Data: **624 762 852 426 197 987 269**
146 415 301 730 78 593

Radix Sort: Example

Data: 624 762 852 426 197 987 269
146 415 301 730 78 593

Bucket	Pass1	Pass2	Pass3
0	730	301	78
1	301	415	146 - 197
2	762 - 852	624 - 426	269
3	593	730	301
4	624	146	415 - 426
5	415	852	593
6	426 - 146	762 - 269	624
7	197 - 987	78	730 - 762
8	78	987	852
9	269	593 - 197	987

By keeping relative order from the previous pass, ties can be broken on subsequent passes

Collision order resolved by first digit ordering

Hash Table Sorting: Radix Sort

- Sorts positive integer values over any range
- Hash table size of 10 (0 through 9)
- Values are hashed according to their least significant digit (the "ones" digit)
- Values then rehashed according to the next significant digit (the tens digit) while keeping their relative ordering
- Process is repeated until we run out of digits

Time Complexity

- K passes (where K is number of digits)
- Each pass puts N elements in buckets
  - O(KN)
- How does this compare to O(NlogN) sorts?

Your Turn

- Complete worksheet #39 where you will simulate radix sort on the following values:
- 742 247 391 382 616 872 453 925 732 142 562