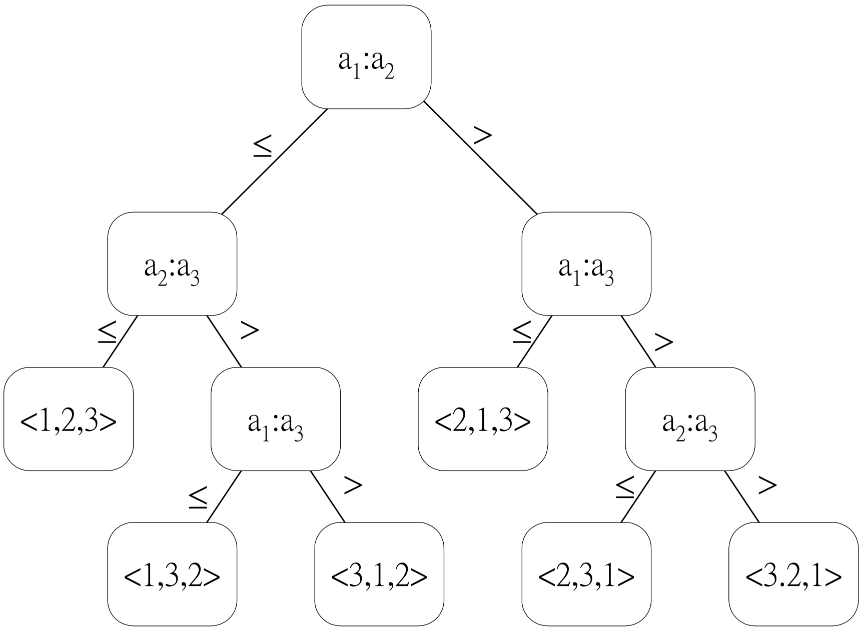
8.Sorting in linear time

Hsu, Lih-Hsing

8.1 Lower bound for sorting

The decision tree model



Theorem 9.1. Any decision tree that sorts n elements has height $\Omega(n \log n)$.

Proof:

$$n! \le l \le 2^h,$$

$$h \ge \log(n!) = \Omega(n \lg n).$$

Corollary 9.2 Heapsort and merge sort are asymptotically optimal comparisons.

8.2 Counting sort

- ***Assume that each of the n input elements is an integer in the range 1 to k for some integer k.***

COUNTING_SORT(A,B,k)

for $i \leftarrow 1$ **to** k

  do $c[i] \leftarrow 0$

for $j \leftarrow 1$ **to** length[A]

  do $c[A[j]] \leftarrow c[A[j]] + 1$

  ▶ $c[i]$ now contains the number of elements equal to i

for $i \leftarrow 2$ **to** k

  do $c[i] \leftarrow c[i] + c[i-1]$

▶ $c[i]$ now contains the number of elements less than or equal to i

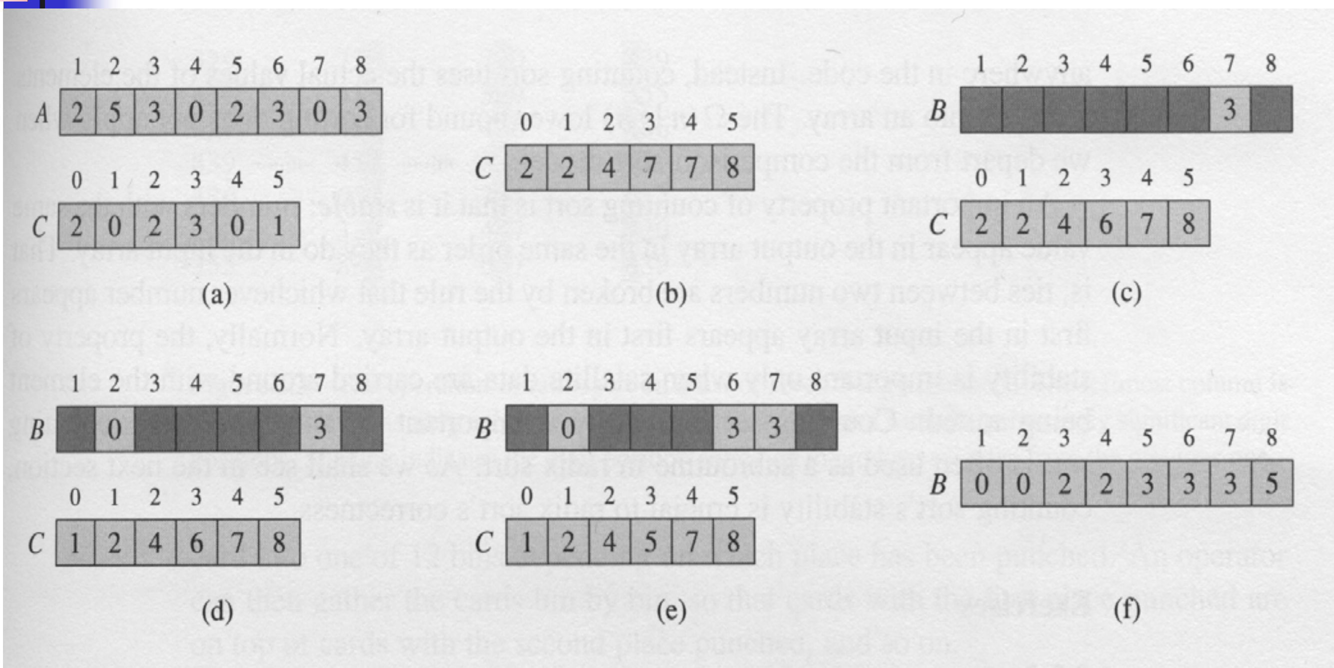
for $j \leftarrow length[A]$ **downto** 1

  do $B[c[A[j]]] \leftarrow A[j]$

    $c[A[j]] \leftarrow c[A[j]] - 1$

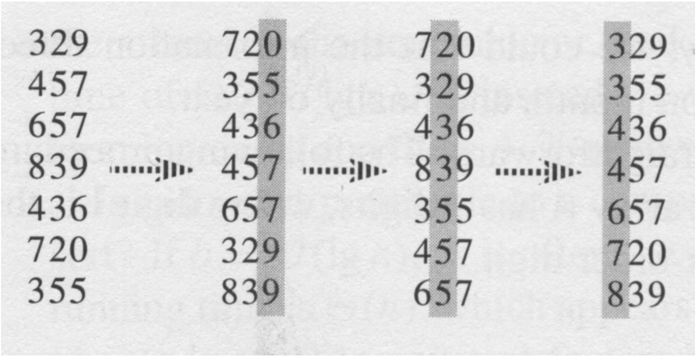
- Analysis: $O(k+n)$
- Special case: $O(n)$ when $k=O(n)$.
- Property: ***stable*** (number with the same value appear in the output array in the same order as they do in the input array.)

The operation of Counting-sort on an input array A[1..8]



8.3 Radix sort

Used by the card-sorting machines you can now find only in computer museum.



RADIX_SORT(A,d)

for $i \leftarrow 1$ **to** d

    do use a stable sort to sort array A on digit i

Analysis: $O(d(n+k)) = O(dn+dk)$

NOTE: not ***in-place*** (in-place: only constant number of elements of the input array are ever stored outside the array.)

Lemma 8.3

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

Lemma 8.4

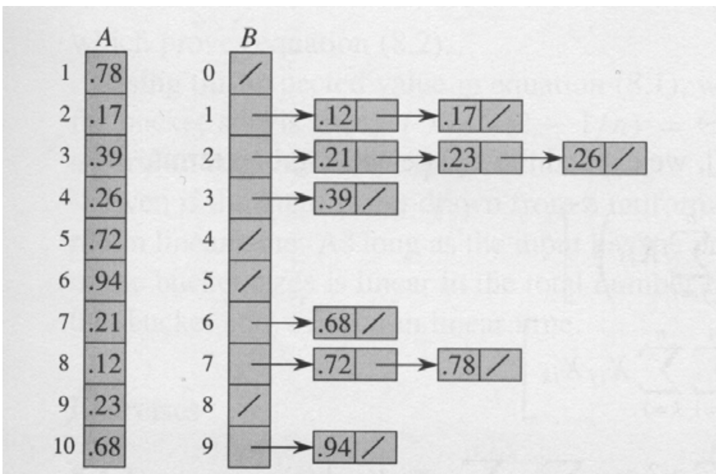
Given n b-bit numbers and any positive integer $r \le b$, RADIX- SORT correctly sorts these numbers in $\Theta((b/r)(n+2^r))$ time.

Proof : Choose $d = \lceil b/r \rceil$.

IF $b < \lfloor \lg n \rfloor$, choose $r = b$.

$\therefore\ \Theta((b/r)(n+2r)) = \Theta(bn/\log n)$

8.4 Bucket sort



BUCKET_SORT(A)

```
n ← length[A]
for i ← 1 to n
    do insert A[i] into
       list B[⌊nA[i]⌋]
for i ← 1 to n-1
    do sort list B[i] with insertion sort
concatenate B[0], B[1], ..., B[n-1] together in
  order
```

Analysis

The running time of bucket sort is

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

taking expectations of both sides and using linearity of expectation, we have

$$E[T(n)] = E\left[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)\right]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} E[O(n_i^2)]$$

$$= \Theta(n) + \sum_{i=0}^{n-1} O[E(n_i^2)]$$

We claim that

$$E[n_i^2] = 2 - 1/n$$

We define indicator random variables
X_{ij} = I {A[*j*] falls in bucket *i*}
for i = 0, 1, …, n-1 and j = 1, 2,…,n. thus,

$$n_i = \sum_{j=1}^{n} X_{ij}.$$

$$E[n_i^2] = E\left[\left(\sum_{j=1}^{n} X_{ij}\right)^2\right]$$

$$= E\left[\sum_{j=1}^{n}\sum_{k=1}^{n} X_{ij}X_{ik}\right]$$

$$= E\left[\sum_{j=1}^{n} X_{ij}^2 + \sum_{1\le j\le n}\ \sum_{\substack{1\le k\le n \\ k\ne j}} X_{ij}X_{ik}\right]$$

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

Indicator random variable X_{ij} is 1 with probability 1/n and 0 otherwise, and therefore

$$E[X_{ij}^2] = 1\cdot\frac{1}{n} + 0\cdot\left(1-\frac{1}{n}\right)$$

$$= \frac{1}{n}$$

When $k \neq j$, the variables X_{ij} and X_{ik} are independent, and hence

$$E[X_{ij}X_{ik}] = E[X_{ij}]E[X_{ik}]$$

$$= \frac{1}{n}\cdot\frac{1}{n}$$

$$= \frac{1}{n^2}.$$

$$E[n_i^2] = \sum_{j=1}^{n} \frac{1}{n} + \sum_{1 \le j \le n} \sum_{\substack{1 \le k \le n \\ k \ne j}} \frac{1}{n^2}$$

$$= n \cdot \frac{1}{n} + n(n-1) \cdot \frac{1}{n^2}$$

$$= 1 + \frac{n-1}{n}$$

$$= 2 - \frac{1}{n}$$

We can conclude that the expected time for bucket sort is $\Theta(n) + n \cdot O(2-1/n) = \Theta(n)$.

Another analysis:

$$\sum_{i=0}^{n-1} O(E[n_i^2]) = O(\sum_{i=0}^{n-1} E[n_i^2]) = O(\sum_{i=0}^{n-1} \Theta(1)) = O(n)$$

Because

$$E(n_i^2) = Var[n_i] + E^2[n_i]$$

$$E[n_i] = np = 1 \quad \text{where } p = \frac{1}{n}$$

$$Var[n_i] = np(1-p) = 1 - \frac{1}{n}$$

$$E(n_i^2) = 1 - \frac{1}{n} + 1^2 = 2 - \frac{1}{n} = \Theta(1)$$

(Basic Probability Theory)