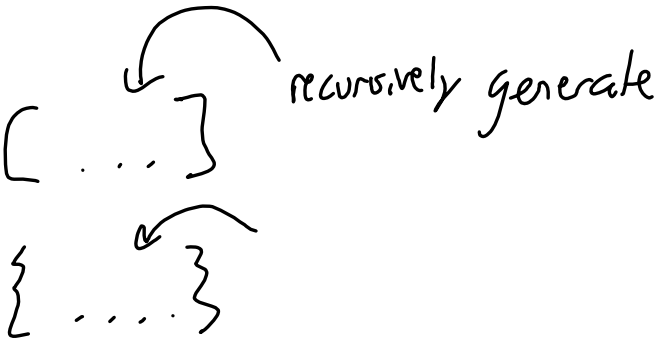



true / false

1234

"a"

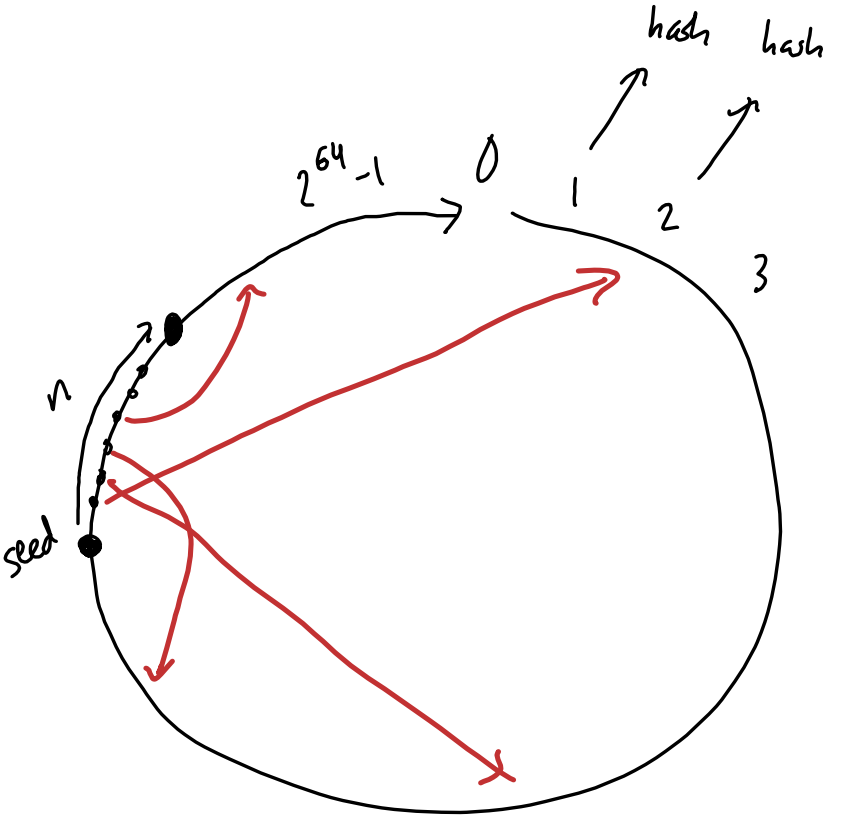
64 bit

seed = int

split(seed, i) =
hash(seed + i)

hash(seed)

seed → seed + 1

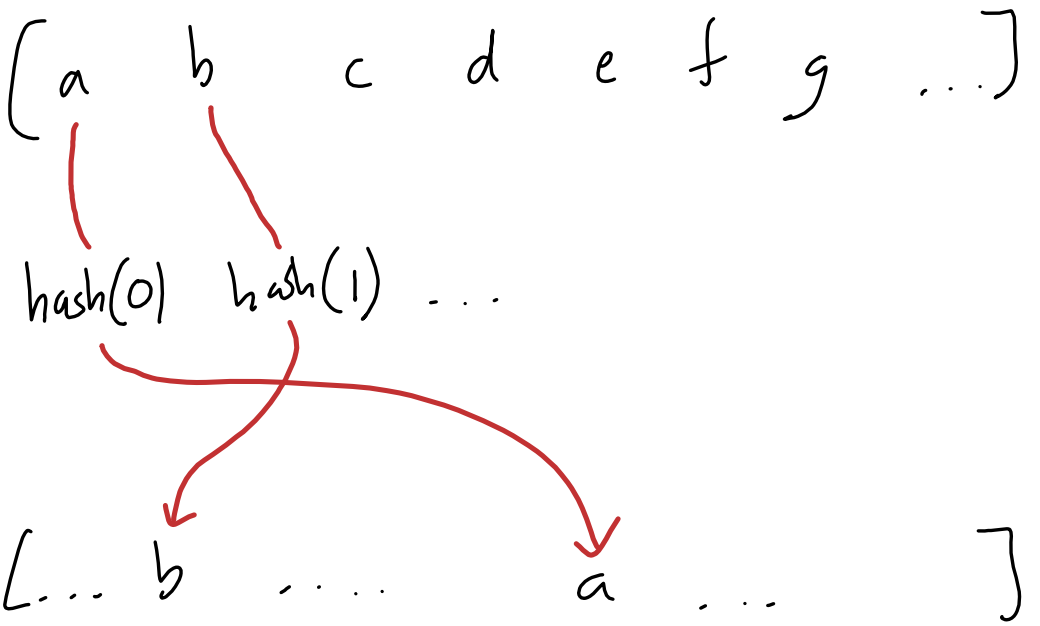


Knuth-Fisher-Yates shuffle

(sequential shuffle)

array A  swap(i, j)

```
for i from 0 to n-2:
    j = random_int(i, n)
    swap(i, j)
```




Could use: Radix Sort

"integer sorting"

$O(K \cdot n)$ work

K = bit width

# Parallel Counting Sort

elem [ a b c d e f ... ]

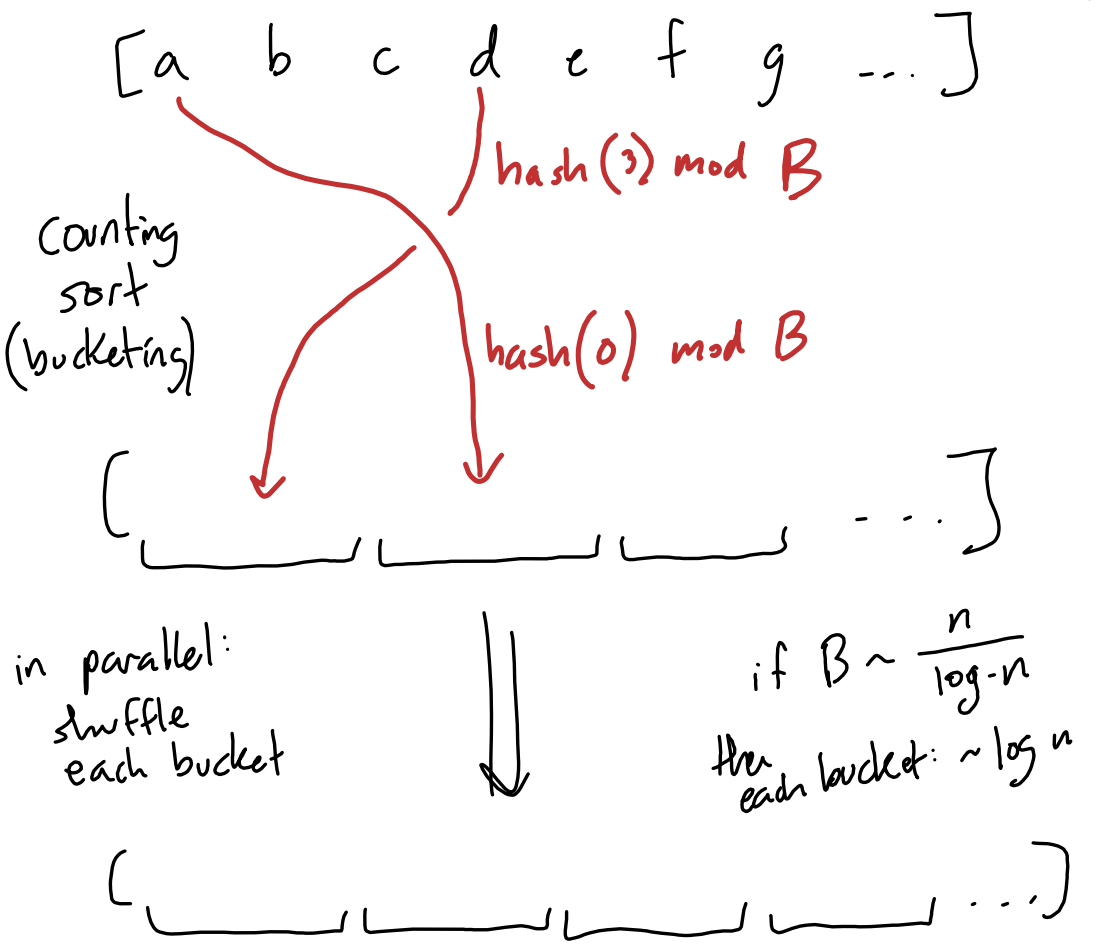
Key [ 10 1 1 2 20 ... ]

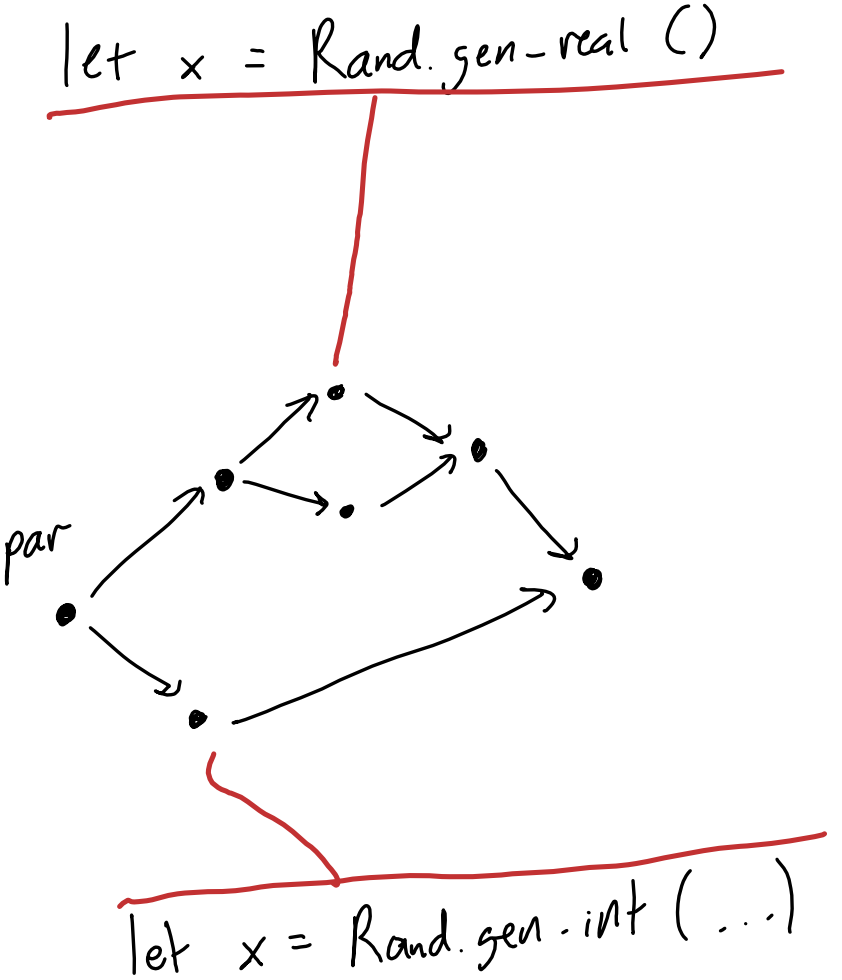
[ Key 0 , Key 1 , Key 2 , ... ]

Building Blocks:

- sequential shuffle
- counting sort (bucket-by-key)
- hash functions, etc

---




let x = Rand.gen_real ()
par
let x = Rand.gen_int (...)

DePa

depths and paths