Randomized Quicksort

- recursive divide and conquer algorithm
- **Input** : set S of n distinct keys
- **Output** : the keys of S in increasing order

```
def rqs(S):
    if size(S) == 0:
        return
    if size(S) == 1:
        return S[0]
    pivot = uniform_random_element(S)
    Sl = set()
    Sr = set()
    for item in S:
        if item < pivot:
            insert(Sl, item)
        else:
            insert(Sr, item)
    return rqs(Sl) + [pivot] + rqs(Sr)
```

Analysis

- keys k_1, k_2 are only compared if one of them is the pivot for their partition
- keys are compared at most once
  - as a result, RQS makes at most $\binom{n}{2}$ comparisons, so it runs in $\mathcal{O}(n^2)$ time
  - worst case actually is $O(n^2)$ time

- if keys end up in different partitions, they are never compared

Expected (average case) running time

Let C be the total number of comparisons

define $c_{ij} = \begin{cases} 0 & \text{if } z_i \text{ not compared with } z_j \\ 1 & \text{otherwise} \end{cases}$

Lemma: $E(c_{ij}) = P(z_i \text{ is compared to } z_j) = \dfrac{2}{j-i+1}$

note that z_i and z_{i+1} are always compared because there is no pivot between them

let $Z_{ij} = z_i, z_{i+1}, \ldots, z_j$

consider the first time p is chosen from Z_{ij} (must happen because $|Z_{ij}| > 1$)

Case 1: $z_i < p < z_j$, so they are separated into different partitions

- don't care, because then z_i and z_j are not compared

Case 2: $z_i = p$ or $z_j = p$

- probability $\dfrac{2}{|Z_{ij}|} = \dfrac{2}{j-i+1}$
- since the p is compared against every value in Z_{ij}, z_i and z_j get compared in this case

Proof of average case running time

$$E(C) = E\left(\sum_{1 \le i < j \le n} c_{ij}\right) = \sum_{1 \le i < j \le n} E(c_{ij})$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \text{ (by lemma)}$$

let $k = j - i$, then

$$= 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k+1}$$

$$\le 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k}$$

$$\leq 2\sum_{i=1}^{n} H_n \text{ (}H_n\text{ is the }n^{\text{th}}\text{ harmonic number)}$$

$$\in \mathcal{O}(n\log(n))$$