クイックセレクト

要素の列が与えられたときに、小さい方から数えて k 番目の要素を見つける問題を考える。クイックソートなどのソートを使うと O(n log n)だが、無駄が多い。k<<n の場合には、ヒープソートを使えば O(n+klogn)で計算が可能。ただし、k~n の場合（たとえば中央値を見つける場合 k=n/2 ）には、O(n log n)かかってしまう。以下に説明する quickselect はクイックソートの原理を応用したアルゴリズムで、平均の場合に O(n)、最悪の場合 O(n^2)で k 番目の要素を見つけることができる。

```
// 配列 A の i 番目から j 番目の要素のうち、小さい方から m 番目の要素を返す。0 始まり。
Quickselect(int[] A, int i, int j, int m){
    if (i==j) return i;
    int q = find_pivot(A, i, j);        // 始めの 2 つの要素のうち大きい方
    int k = partition(A, i, j);         // i~k-1 に q より小さい要素、k~j に q 以上の要素
    if (m< k)    return Quickselect(A, i, k-1, m);   // 前半に含まれている。
    else         return Quickselect(A, k, j, m-k);   // 後半に含まれている。
}
```

計算の手間について、最善の場合は、partition で半分ずつに分かれていくとして、n+n/2+n/4+n/8+…+1 < 2n で O(n)。最悪の場合には、partition で 1:n-1 に分かれていくとして、n+(n-1)+(n-2)+….1 < n^2/2 で O(n^2)。平均の場合については、O(n) になる。その導出は以下の通りである。

再帰方程式を立てる。i 個と n-i 個に分かれるが、上限を見積もりたいので、 大きい方に分岐すると仮定する。
T(n) < (n-1) + 2/n * sum_{n/2..n-1} T(i)

T(n)< 4n と仮定して、帰納法で示す。
n = 2 のとき T(2) = 1<8 なので成立。
n = 1..k-1 で成立とする。
T(k) = (k-1) + 2/k * sum{k/2..k-1}T(i)
 < (k-1) + 2/k * sum{k/2..k-1}4i
 = (k-1) + 8/k * sum{k/2..k-1}4i
 = (k-1) + 8/k * 3/8kk ※より
 = (k-1) + 3k
 < 4k

※ (k + k/2)*k/2/2 = (3/2)*k*k/4 = 3/8kk