WIKIPEDIA

Heap's algorithm

Heap's algorithm generates all possible permutations of n objects. It was first proposed by B. R. Heap in 1963.[1] The algorithm minimizes movement: it generates each permutation from the previous one by interchanging a single pair of elements; the other $n-2$ elements are not disturbed. In a 1977 review of permutation-generating algorithms, Robert Sedgewick concluded that it was at that time the most effective algorithm for generating permutations by computer.[2]

The sequence of permutations of n objects generated by Heap's algorithm is the beginning of the sequence of permutations of $n+1$ objects. So there is one infinite sequence of permutations generated by Heap's algorithm (sequence A280318 in the OEIS).

Details of the algorithm

Suppose we have a permutation containing n different elements. Heap found a systematic method for choosing at each step a pair of elements to switch, in order to produce every possible permutation of these elements exactly once. Let us describe Heap's method in a recursive way. First we set a counter i to 0. Now we perform the following steps repeatedly until i is equal to n. We use the algorithm to generate the $(n-1)!$ permutations of the first $n-1$ elements, adjoining the last element to each of these. This generates all of the permutations that end with the last element. Then if n is odd, we switch the first element and the last one, while if n is even we can switch the ith element and the last one (there is no difference between n even and odd in the first iteration). We add one to the counter i and repeat. In each iteration, the algorithm will produce all of the permutations that end with the element that has just been moved to the "last" position. The following pseudocode outputs all permutations of a data array of length n.

```
procedure generate(n : integer, A : array of any):
    if n = 1 then
          output(A)
    else
        for i := 0; i < n - 1; i += 1 do
            generate(n - 1, A)
            if n is even then
                swap(A[i], A[n-1])
            else
                swap(A[0], A[n-1])
            end if
        end for
        generate(n - 1, A)
    end if
```

One can also write the algorithm in a non-recursive format.[3]

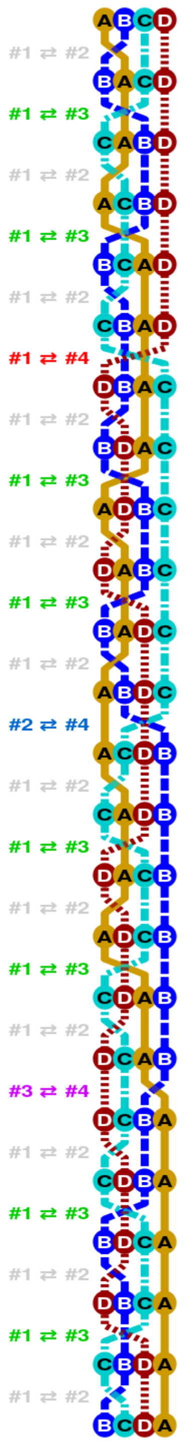
```
procedure generate(n : integer, A : array of any):
    c : array of int

    for i := 0; i < n; i += 1 do
        c[i] := 0
    end for

    output(A)

    i := 0;
    while i < n do
        if  c[i] < i then
```

```
            if i is even then
                swap(A[0], A[i])
            else
                swap(A[c[i]], A[i])
            end if
            output(A)
            c[i] += 1
            i := 0
        else
            c[i] := 0
            i += 1
        end if
    end while
```



A map of the 24 permutations and the 23 swaps used in Heap's algorithm permuting the four letters A (amber), B (blue), C (cyan) and D (dark red)

See also

- Steinhaus–Johnson–Trotter algorithm

References

1. Heap, B. R. (1963). "Permutations by Interchanges" (http://comjnl.oxfordjournals.org/content/6/3/293.full.pdf) (PDF). *The Computer Journal*. **6** (3): 293–4. doi:10.1093/comjnl/6.3.293 (https://doi.org/10.1093/comjnl/6.3.293).
2. Sedgewick, R. (1977). "Permutation Generation Methods". *ACM Computing Surveys*. **9** (2): 137–164. doi:10.1145/356689.356692 (https://doi.org/10.1145/356689.356692).
3. Sedgewick, Robert. "a talk on Permutation Generation Algorithms" (http://www.cs.princeton.edu/~rs/talks/perms.pdf) (PDF).