Range Minimum Query (RMQ) Examples

1. Recursive solution (`rmq`)
2. Memoized solution (`rmq_m`)
3. Iterative solution (`rmq_i`)

In [167]:
```
import random
```

In [168]:
```
# Problem size / array length
N = 1000

# Generate some random data (positive integers)
A = []
for i in range(N):
    A.append(random.randint(1,5*N))
```

In [169]:
```
# Recursive RMQ.  Exponential running time!!
def rmq(A, i, j):
    if i == j:
        return A[i]
    else:
        return min(rmq(A,i,j-1), rmq(A,i+1,j))
```

In [170]:
```
# Memoized RMQ.  O(n^2) running time.
def rmq_m(A, i, j, T):
    if i == j:
        return A[i]
    elif T[i][j] > 0:
        return T[i][j]
    else:
        T[i][j] = min(rmq_m(A,i,j-1, T), rmq_m(A,i+1,j, T))
        return T[i][j]
```

In [171]:
```
# Iterative RMQ.  O(n^2) running time.
def rmq_i(A, i, j, T):
    n = len(A)
    for k in range(n):
        T[k][k] = A[k]
    for k in range(1,n):
        for l in range(0,n-k):
            T[l][k+l] = min(T[l][k+l-1], T[l+1][k+l])
    return T[i][j]
```

In [172]:
```
# NxN "matrix" initialized to zero
# Used with memoized and iterative solutions to store memoization / pre-computed queries
T = [ [ 0 for j in range(N)] for i in range(N)]
```

In [173]:
```
# This takes a few seconds...
rmq(A, 40, 62)
```

Out[173]: 449

In [174]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
rmq_m(A, 0, N-1, T)
```

Out[174]: 5

In [175]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
rmq_i(A, 0, N-1, T)
```

Out[175]: 5

In [176]:
```
# Now queries are just look-ups in T
print(T[45][79])
print(T[517][623])
```

```
314
5
```

Timings for Memoized RMQ

In [177]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
%time rmq_m(A,1,100,T)
```

```
CPU times: user 5.05 ms, sys: 82 µs, total: 5.13 ms
Wall time: 5.49 ms
```

Out[177]: 143

In [178]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
%time rmq_m(A,1,200,T)
```

```
CPU times: user 16 ms, sys: 32 µs, total: 16 ms
Wall time: 16 ms
```

Out[178]: 143

In [179]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
%time rmq_m(A,1,400,T)
```

```
CPU times: user 69.3 ms, sys: 270 µs, total: 69.6 ms
Wall time: 70.2 ms
```

Out[179]: 23

In [180]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
%time rmq_m(A,1,800,T)
```

```
CPU times: user 293 ms, sys: 1.08 ms, total: 294 ms
Wall time: 296 ms
```

Out[180]: 5

Timings for Iterative RMQ

In [181]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
%time rmq_i(A,1,100,T)
```

```
CPU times: user 281 ms, sys: 895 µs, total: 282 ms
Wall time: 283 ms
```

Out[181]: 143

In [182]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
%time rmq_i(A,1,200,T)
```

```
CPU times: user 290 ms, sys: 2.11 ms, total: 292 ms
Wall time: 296 ms
```

Out[182]: 143

In [183]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
%time rmq_i(A,1,400,T)
```

```
CPU times: user 282 ms, sys: 1.08 ms, total: 283 ms
Wall time: 286 ms
```

Out[183]: 23

In [184]:
```
T = [ [ 0 for j in range(N)] for i in range(N)]
%time rmq_i(A,1,800,T)
```

```
CPU times: user 288 ms, sys: 511 µs, total: 289 ms
Wall time: 290 ms
```

Out[184]: 5