Додоаток 2: Оптимизиран алгоритам за пресметување на медијана

```
/*
* This Quickselect routine is based on the algorithm described in
* "Numerical recipes in C", Second Edition,
* Cambridge University Press, 1992, Section 8.5, ISBN 0-521-43108-5
* This code by Nicolas Devillard - 1998. Public domain.
*/
#define ELEM_SWAP(a,b) { register u8 t=(a);(a)=(b);(b)=t; }
static u8 quick_s(u8 arr[], u32 n)
{
  int low, high ;
  int median;
  int middle, ll, hh;
  low = 0 ; high = n-1 ; median = (low + high) / 2;
  for (;;) {
    if (high <= low) /* One element only */
    return arr[median] ;

    if (high == low + 1) { /* Two elements only */
      if (arr[low] > arr[high])
      ELEM_SWAP(arr[low], arr[high]) ;
      return arr[median] ;
    }
  /* Find median of low, middle and high items; swap into position low */
  middle = (low + high) / 2;
  if (arr[middle] > arr[high]) ELEM_SWAP(arr[middle], arr[high]) ;
  if (arr[low] > arr[high]) ELEM_SWAP(arr[low], arr[high]) ;
  if (arr[middle] > arr[low]) ELEM_SWAP(arr[middle], arr[low]) ;
  /* Swap low item (now in position middle) into position (low+1) */
  ELEM_SWAP(arr[middle], arr[low+1]) ;
  /* Nibble from each end towards middle, swapping items when stuck */
  ll = low + 1;
  hh = high;
  for (;;) {
    do ll++; while (arr[low] > arr[ll]) ;
    do hh--; while (arr[hh] > arr[low]) ;
    if (hh < ll)
      break;
    ELEM_SWAP(arr[ll], arr[hh]) ;
  }
  /* Swap middle item (in position low) back into correct position */
  ELEM_SWAP(arr[low], arr[hh]) ;
  /* Re-set active partition */
  if (hh <= median) low = ll;
  if (hh >= median) high = hh - 1;
  }
```

}