S. Schwoon / D. Suwimonteerabuth Winter Term 2008/09

Fundamental Algorithms

Solution Keys 3

1. (Not available)

2. **Procedure** `Hanoi`(n, i, j)

    if $m > 0$ **then**
        `Hanoi`$(m-1, i, 6-i-j)$;
        `move`(i, j);
        `Hanoi`$(m-1, 6-i-j, j)$;
    fi

    Let $t(m)$ be the number of calls to `move` in order to solve the puzzle with m disks. It can be seen from the algorithm that

    $$t(m) = \begin{cases} 0\ , & \text{if } m = 0 \\ 2t(m-1)+1\ , & \text{otherwise}\ , \end{cases}$$

    which is easy to prove by induction that $t(m) = 2^m - 1$.

3. (a) **Procedure** `merge`(u, v)

    Input: sorted arrays u and v of lengths m and n, respectively
    Output: sorted array of length $m+n$, containing elements from u and v

    $i, j, k := 1$;
    while $i \le m$ **and** $j \le n$ **do**
        if $u[i] < v[j]$ **then**
            $s[k] := u[i]$;
            $i := i + 1$;
        else
            $s[k] := v[j]$;
            $j := j + 1$;
        fi
        $k := k + 1$;
    od
    while $i \le m$ **do**
        $s[k] := u[i]$;
        $k := k + 1$; $i := i + 1$;
    od
    while $j \le n$ **do**
        $s[k] := v[j]$;
        $k := k + 1$; $j := j + 1$;
    od
    return s;

(b) **Procedure** `mergesort`(a)

Input: array a of length n
Output: the array a in sorted order

```
if n ≤ 1 then
    return a;
fi
m := ⌊n/2⌋;
for i = 1 to m do
    u[i] = a[i];
end
for i = m + 1 to n do
    v[i − m] = a[i];
end
return merge(mergesort(u), mergesort(v));
```

Let $t(n)$ be the time required by `mergesort` to sort an array of length n.

$$t(n) = t(\lfloor n/2 \rfloor) + t(\lceil n/2 \rceil) + g(n) \ ,$$

where $g(n) \in \Theta(n)$. For simplicity assume that $n = 2^k$, for some $k \geq 0$ (we skip the general case). There must exist a constant c such that

$$\begin{aligned} t(n) &\leq 2t\left(\frac{n}{2}\right) + cn \\ &\leq 2\left(2t\left(\frac{n}{4}\right) + c\frac{n}{2}\right) + cn \ = \ 4t\left(\frac{n}{4}\right) + cn + cn \\ &\vdots \\ &\leq 2^k t(1) + \underbrace{cn + \ldots + cn}_{= \, cn \lg n} \end{aligned}$$

Therefore, $t(n) \in \mathcal{O}(n \log n)$.