アルゴリズム及び演習 第 4 回 補足

小野 孝男

2007 年 5 月 21 日

問題 1 について

リスト操作 (や二分探索木の操作) では, 要素に対する「ポインタのポインタ」を使うと便利なことがある. 例えば, 「指定した値を持つ要素を削除する」関数は「ポインタのポインタ」を使って図 1 のように書くことができる.

```
void remove(struct element *l, char data)
{
    struct element **p = &l->next;
    while (*p != NULL) {
        if ((*p)->data == data) {
            *p = (*p)->next;
            break;
        }
        p = &(*p)->next;
    }
}
```

図 1 「ポインタのポインタ」を使って要素を削除する関数

これを使えば, 先頭に入れていたダミーの要素がなくても同じオーダーの時間で処理をすることが可能である.

問題 4 の別解

n 個の頂点を $v_1, v_2, \ldots, v_n$ とおき, n に関する帰納法で証明する.

$n = 2$ のときは存在する辺が (v_1, v_2) または (v_2, v_1) のいずれかであり, どちらの場合に

おいてもハミルトン路が存在する.

$n \leq k$ で命題が成り立つと仮定して $n = k + 1$ のときを考える. ここで, $v = v_{k+1}$ に対し「v に入る辺を持つ頂点の集合」V_O,「v から出る辺を持つ頂点の集合」V_I を考える. 元のグラフ G に対し, V_O 及び V_I で誘導される部分グラフをそれぞれ G_O, G_I とおく. G_O, G_I はいずれも頂点数が高々 k である. つまり, 帰納法の仮定から G_O, G_I はどちらもハミルトン路を持つ.

$V_I = \emptyset$ のときは $v_1, v_2, \ldots, v_n$ のいずれからも v に入る辺を持つ. よって, G_O のハミルトン路の最後に v_{k+1} を加えたものは G に対するハミルトン路である. 逆に $V_O = \emptyset$ のときは $v_1, v_2, \ldots, v_n$ のいずれへも v_{k+1} から行くことができる. 従って G_I のハミルトン路の先頭に v_{k+1} を追加すれば G のハミルトン路が得られる.

最後に V_I, V_O のいずれも空でない場合を考える. ここで G_I のハミルトン路 P_I と G_O のハミルトン路 P_O をとる. これらに対し,「P_I を全てたどり, v_{k+1} を経由して P_O を全てたどる」経路 P を考えるとこれは G の全ての頂点をちょうど一度ずつ通っているので G のハミルトン路である.