Γλώσσες Προγραμματισμού Ι

Λύσεις των προγραμματιστικών θεμάτων της επαναληπτικής εξέτασης (Σεπτεμβρίου 2020).

Θέμα 4

1. Αποδοτική λύση $O(N \log N)$:

```
fun bidlist kvs =
  let fun walk [] next acc = List.rev acc
        | walk ((k, v) :: rest) next acc =
            if k = next
            then walk rest (next + 1) (v :: acc)
            else if k = next - 1
            then walk rest next ((v + hd acc) :: tl acc)
            else walk ((k, v) :: rest) (next + 1) (0 :: acc)
      fun keycomp ((k1, v1), (k2, v2)) = k1 > k2
  in  walk (ListMergeSort.sort keycomp kvs) 0 []
  end
```

2. Λιγότερο αποδοτική λύση $O(N^2)$:

```
fun bidlist [] = []
  | bidlist ((k, v) :: rest) =
      let fun insert 0 v [] = [v]
            | insert k v [] = 0 :: insert (k-1) v []
            | insert 0 v (x :: r) = (x+v) :: r
            | insert k v (x :: r) = x :: insert (k-1) v r
      in  insert k v (bidlist rest)
      end
```

3. Συνηθισμένα λάθη:

   - Προσθήκη στοιχείων στο τέλος της λίστας (π.χ., με `@ [x]`), που χαλάει την πολυπλοκότητα.
   - Κατασκευή της λίστας σε αντίστροφη σειρά.
   - Να μην προστίθενται μηδενικά για όσα κλειδιά δεν υπάρχουν.
   - Χρήση array.

Θέμα 5α

1. Λύση:

```
gcd(X,0,G) :- gt(X,0), G = X.
gcd(0,X,G) :- gt(X,0), G = X.
gcd(X,Y,G) :- gt(Y,0), ge(X,Y), plus(Y,X1,X), gcd(X1,Y,G).
gcd(X,Y,G) :- gt(X,0), gt(Y,X), plus(X,Y1,Y), gcd(X,Y1,G).

ge(X,0) :- natural_number(X).
ge(s(X),s(Y)) :- ge(X,Y).

gt(s(X),0) :- natural_number(X).
gt(s(X),s(Y)) :- gt(X,Y).

plus(0,X,X) :- natural_number(X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).

natural_number(0).
natural_number(s(X)) :- natural_number(X).
```

2. Συνηθισμένα λάθη:

   - Μετατροπή σε ακέραιους αριθμούς.
   - Συγκρίσεις όρων στην παραπάνω αναπαράσταση με χρήση αριθμητικών τελεστών.

Θέμα 5β

1. Λύση:

```
/* odd_permutation(Xs, Ys) is true if Ys is an odd permutation of Xs.      */
odd_permutation(Xs, Ys):-
  permute(Xs, Ys),
  sign_of_product_of_differences(Xs, 1, D),
  sign_of_product_of_differences(Ys, 1, E),
  D =\= E.

sign_of_product_of_differences([], D, D).
sign_of_product_of_differences([Y|Xs], D0, D):-
  sign_of_product_of_differences_1(Xs, Y, D0, D1),
  sign_of_product_of_differences(Xs, D1, D).

sign_of_product_of_differences_1([], _, D, D).
sign_of_product_of_differences_1([X|Xs], Y, D0, D):-
  Y =\= X,
  D1 is D0 * (Y - X) // abs(Y - X),
  sign_of_product_of_differences_1(Xs, Y, D1, D).

/* permute(Xs, Ys) is true if Ys is a permutation of the list Xs.          */
permute([], []).
permute([X|Xs], Ys1):-
  permute(Xs, Ys),
  select(X, Ys1, Ys).
```

2. Συνηθισμένα λάθη:
   - Πρόγραμμα που επιστρέφει μία/κάποιες λύσεις και μετά πάει σε infinite loop.

Θέμα 6

1. Αποδοτική λύση $O(N)$ χρόνο και χώρο:

```
def countsumk_ON(A, K):
    seen = {0: 1}
    sum = result = 0
    for x in A:
        sum += x
        if sum - K in seen:
            result += seen[sum - K]
        if sum in seen:
            seen[sum] += 1
        else:
            seen[sum] = 1
    return result
```

2. Λιγότερο αποδοτική λύση $O(N^2)$ χρόνο, $O(1) χώρο:

```
def countsumk_ON2(A, K):
    N = len(A)
    result = 0
    for i in range(N):
        sum = 0
        for j in range(i, N):
            sum += A[j]
            if sum == K: result += 1
    return result
```

3. Ακόμα λιγότερο αποδοτική λύση $O(N^3)$ (το `sum` και τα slices κοστίζουν!) και $O(N)$ χώρο (τα slices λιστών αντιγράφουν τα στοιχεία):

```
def countsumk(A, K):
    N = len(A)
    result = 0
    for i in range(N):
        for j in range(i, N):
            if sum(A[i:j+1]) == K: result += 1
    return result
```

4. Συνηθισμένα λάθη:

    - Δεν προσμετρώνται suffixes με άθροισμα `K`.
    - Η greedy λύση με δύο δείκτες που δουλεύει μόνο για μη αρνητικούς αριθμούς πήρε σχεδόν 100% της βαθμολογίας.