Tema 1 – Decodor Morse

Andrei Olaru & Tudor Berariu

IA 2016

1 Programare dinamică

Programarea dinamică este o metodă de rezolvare a problemelor și de optimizare a soluțiilor, prin descompunerea problemei în subprobleme și asamblarea soluțiilor optime ale subproblemelor în soluția optimă pentru problema inițială.

Programarea dinamică este utilizată în diverși algoritmi de inteligență artificială, de exemplu în învățarea automată, planificare, găsirea căilor optime, algoritmi pe grafuri, etc.

O problemă poate fi rezolvată utilizând programarea dinamică doar dacă aceasta are proprietatea substructurii optime. Mai precis, pentru a aplica programare dinamică o problemă trebuie să fie descompusă în subprobleme care se suprapun.

Spre deosebire de strategia divide-et-impera în cazul programării dinamice subproblemele se suprapun. De aceea, algoritmii de programare dinamică utilizează memorie suplimentară pentru stocarea rezultatelor intermediare.

Două exemple de probleme a căror descompunere nu duce la subprobleme ce se suprapun ar fi: aflarea maximului unui vector sau calculul factorialului unui număr. Spre deosebire de acestea, aflarea unui termen din șirul lui Fibonacci conduce la subprobleme ce depind de calcule comune (alte subprobleme), la fel și calculul celei mai lungi sub-secvențe crescătoare a unui vector.

Ideea de bază în programarea dinamică este ca fiecare subproblemă să fie rezolvată o singură dată, iar rezultatul ei să fie memorat pentru a fi folosit de oricâte ori se mai ajunge la acea subproblemă.

Există două abordări în programarea dinamică: ascendentă (bottom-up) și descendentă (top-down) pe care le vom exemplifica cu cele două probleme alese ca exemplu anterior.

1.1 Programare dinamică descendentă (memoizare)

Numele spune tot.

```
mem = { 0: 1, 1: 1 }

def fibonacci(n):
    if n not in mem:
        mem[n] = fibonacci(n-1) + fibonacci(n-2)
    return mem[n]
```

1.2 Programare dinamică ascendentă

Se folosește de obicei atunci când rezolvăm incremental o problemă de la un caz de bază către altele mai generale. Să luăm exemplul celei mai lungi sub-secvențe crescătoare.

```
def longest_increasing_subsequence(l):
```

```
# Memorăm soluțiile pe parcurs
lis  = [1 for _ in l]   # lungimea celei mai lungi subsecvențe
back = [-1 for _ in l]  # elementul anterior

for j in range(len(l)):
    for i in range(i):
        if l[j] > l[i] and lis[j] < lis[i] + 1:
            # Am găsit o subsecvență mai lungă pentru j prin i
            lis[j] = lis[i] + 1
            back[j] = i

# Căutăm elementul care încheie cea mai lungă subsecvență
best_length, best_idx = 1, 0
for idx, length in enumerate(lis):
    if length > best_length:
        best_length, best_idx = length, idx

# Reconstruim soluția
idx = best_idx
solution = []
while idx >= 0:
    solution = [l[idx]] + solution
    idx = back[idx]

return solution
```

1.3 Resurse

http://web.stanford.edu/class/cs97si/04-dynamic-programming.pdf

2 Decodor Morse

Problema de rezolvat pentru temă este următoarea: se dă un mesaj în cod Morse ca șir de puncte și linii, dar fără spațiere între caractere sau cuvinte. Se dau de asemenea codurile Morse pentru cele 26 de litere din limba engleză și un dicționar de cuvinte care ar putea să apară în mesaj.

Se cer toate interpretările posibile ale mesajului dat. O interpretare este o succesiune de cuvinte din dicționar a căror codificare în cod Morse rezultă în exact mesajul dat ca intrare.

De exemplu, mesajul `-.-` poate fi interpretat atât ca litera `K`, cât și ca succesiunile `TET`, `TA` și `NT`. Dacă în dicționar există cuvintele `K` și `TET` (dar nu și `TA` sau `NT`), atunci există două interpretări ale mesajului.

3 Intrare / Ieșire

Implementați funcția `decode(alphabet, inputCode, dictionary)`. Parametrii funcției sunt următorii:

- `alphabet` - un dicționar care are literele limbii engleze drept chei și ca valori șiruri de linii și puncte. De exemplu, pentru cheia `A` avem valoarea `.-` care este codul Morse pentru litera A;
- `inputCode` este un șir de caractere format doar din puncte și linii;
- `dictionary` este o listă de cuvinte formate din literele A-Z.

Funcția `decode` trebuie să întoarcă o listă de posibile interpretări (sau soluții). Fiecare soluție va fi dată ca o listă de cuvinte din dicționar, în ordinea în care apar în mesaj.