Uvod u računarske algoritme

Dinamičko programiranje

Najduža zajednička podsekvenca

Najduža zajednička podsekvenca (LCS)

- Problem pronalaska najduže zajedničke sekvence unutar dva stringa
- Npr. string **AAAG** se smatra podsekvencom za string **CGATAATTGAGA** jer se karakteri **AAAG** nalaze jedan nakon drugog u ovom stringu (ne moraju biti kontinualno jedan iza drugog)

- CG***A***T***AA***TT***G***AGA

- Za dva niza karaktera X=x[0]x[1]...x[n-1] i Y=y[0]y[1]...y[m-1] algoritam traži najduži string S takav da je on podsekvenca kako za X tako i za Y.

Najduža zajednička podsekvenca (LCS)

- Primjena:
  - `diff` program na UNIX operativnim sistemima
  - analiza i poređenje DNK lanaca
  - detekcija plagijata
  - programi za *screen sharing* i *remote control*
- Rekurzivna implementacija
  - brute force ili backtracking bez memoizacije
  - brute force sa memoizacijom
- Iterativna implementacija
  - dinamičko programiranje

Najduža zajednička podsekvenca (LCS)

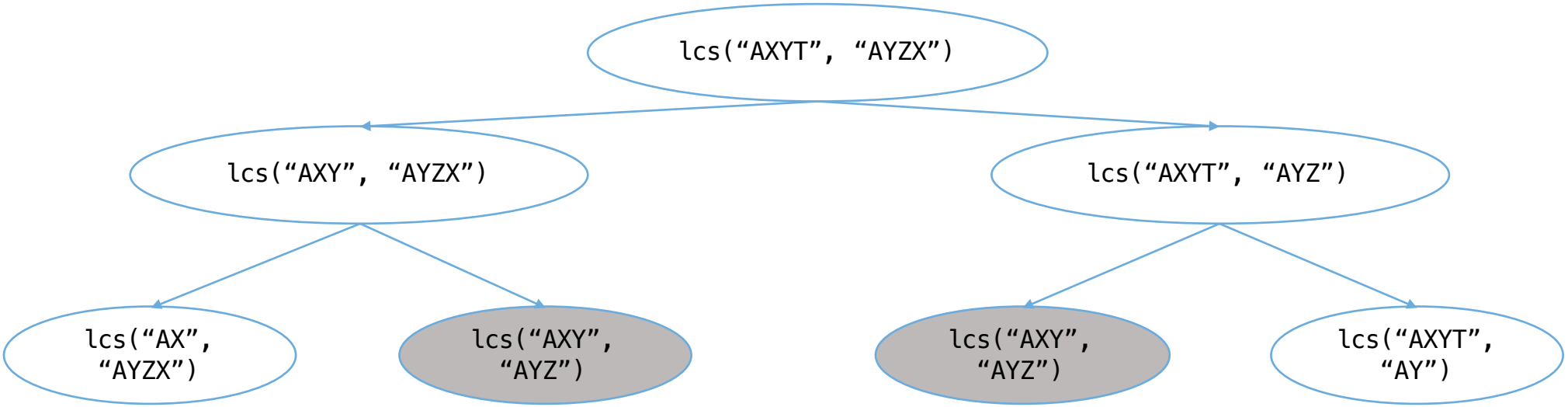
- Rekurzivna implementacija

```
int lcsR(char* a, char* b, int na, int nb){
    if (na == 0 || nb == 0)
     return 0;
   if (a[na-1] == b[nb-1])
     return 1 + lcsR(a, b, na-1, nb-1);
   else
     return max(lcsR(a, b, na, nb-1), lcsR(a, b, na-1, nb));
}
```

- Ponavljanje izračuna istih podsekvence

Najduža zajednička podsekvenca (LCS)

- Rekurzivna implementacija (za stringove AXYT i AYZX):



Najduža zajednička podsekvenca (LCS)

- Rekurzivna implementacija sa memoizacijom

```
int lcsR(char* a, char* b, int na, int nb){
    int sol1, sol2, sol3;
    if(na==0 || nb==0) return 0;
    if(L[na-1][nb-1]>=0)
        return L[na-1][nb-1];
    if(a[na-1]==b[nb-1]){
        sol1=lcsR(a, b, na-1, nb-1);
        L[na-1][nb-1]=sol1+1;
        return sol1+1;
    } else{
        sol2=lcsR(a, b, na-1, nb);
        sol3=lcsR(a, b, na, nb-1);
        L[na-1][nb-1]=max(sol2, sol3);
        return max(sol2, sol3);
    }
}
```

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""									
"A"									
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0								
"A"									
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0							
"A"									
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0						
"A"									
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"									
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0								
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0								
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1							
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1							
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1							
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1						
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1						
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1						
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1					
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"									
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0								
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0								
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0								
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1							
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"									
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0								
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0								
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1							
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1							
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1							
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1						
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1						
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2					
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2					
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2					
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2				
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"									
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1						
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1						
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2					
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2		
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2	3	
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2	3	
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2	3	
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2	3	3
"A"									
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2	3	3
"A"	0	1	1	2	2	2	2	3	3
"H"									

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2	3	3
"A"	0	1	1	2	2	2	2	3	3
"H"	0	1	1	2	2	2	2	3	

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2	3	3
"A"	0	1	1	2	2	2	2	3	3
"H"	0	1	1	2	2	2	2	3	4

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)
  - cilj je smanjiti upotrebu rekurzije (alociranje memorije na stacku pri svakom rekurzivnom pozivu)
  - formirati *lookup* tabelu koja sadrži informacije o najdužoj zajedničkoj podsekvenci za svaku podsekvencu u oba stringa

	""	"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"
""	0	0	0	0	0	0	0	0	0
"A"	0	1	1	1	1	1	1	1	1
"X"	0	1	1	1	1	1	1	1	1
"C"	0	1	1	2	2	2	2	2	2
"G"	0	1	1	2	2	2	2	3	3
"A"	0	1	1	2	2	2	2	3	3
"H"	0	1	1	2	2	2	2	3	4

Najduža zajednička podsekvenca (LCS)

- Iterativna implementacija (dinamičko programiranje)

```
int lcs(char* a, char* b, int na, int nb){
    for(int i=0; i<na; i++)
        L[i][0]=0;
    for(int i=0; i<nb; i++)
        L[0][i]=0;
    for(int i=1; i<=na; i++){
        for(int j=1; j<=nb; j++){
        if(a[i-1]==b[i-1])
            L[i][j]=L[i-1][j-1]+1;
        else{
            L[i][j]=max(L[i-1][j], L[i][j-1]);
        }
      }
    }
    return L[na][nb];
}
```

Primjer 2

- Za dati 2D niz ključeva i vrijednosti dizajnirati i implementirati optimalni algoritam koji vraća maksimalni mogući zbir vrijednosti za koje ukupan zbir ključeva ne prelazi unaprijed definisani limit.
- Naprimjer, za niz:

ključ	2	3	5
vrijednost	8	11	12

- i limit L=6 algoritam treba da vrati broj 19 (zbir vrijednosti prva dva elementa, zbir ključeva je 5 što je manje od zadanog limita)

Primjer 2

- Rekurzivna implementacija

```
int problem2(int limit, int* vrijednosti, int* kljucevi, int broj){
    if(limit==0 || broj==0)
        return 0;
    if(kljucevi[broj-1]>limit){
        return problem2(limit, vrijednosti, kljucevi, broj-1);
    } else{
        return max(vrijednosti[broj-1] + problem2(limit-kljucevi[broj-
1], vrijednosti, kljucevi, broj-1), problem2(limit, vrijednosti,
kljucevi, broj-1));
    }
}
```

Primjer 2

ključ	2	3	5
vrijednost	8	11	12

- Dinamičko programiranje
  - formirati *lookup* tabelu (n, L) koja će sadržavati rješenja podproblema (popunjavati po *bottom-up* principu)

j (max. suma ključeva)

i

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	0	0	0	0	0	0
1	0	0	8	8	8	8	8
2	0	0	8	11	11	19	19
3	0	0	8	11	11	19	**19**

Primjer 2

- Dinamičko programiranje, implementacija:

```
int problem2(int limit, int* vrijednosti, int* kljucevi, int broj){
    int A[broj+1][limit+1];
    for (int i = 0; i <= broj; i++){
        for (int j = 0; j <= limit; j++){
            if (i == 0 || j == 0)
                A[i][j] = 0;
            else if (kljucevi[i - 1] <= j)
                A[i][j] = max(vrijednosti[i - 1] + A[i - 1][j -
kljucevi[i - 1]], A[i - 1][j]);
            else
                A[i][j] = A[i - 1][j];
        }
    }
return A[broj][limit];
}
```