طراحی الگوریتم ها

۷ مهر

ملکی مجد

معرفی درس

- mmalekimajd@gmail.com
- Room 307

- IUST_DA98991

- References
  - Books!
  - Google
  - Prof. and TAs

CLRS	Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
JeffE	Algorithms by Jeff Erickson

- نمره دهی
  - میان ترم و پایان ترم ۱۴. تمرین ها و پروژه ۴. مشارکت کلاسی ۲ (درصد نمره تمرین ها با توجه به تعداد آن ها ممکن است تغییر کند)

فقط سرفصل ها در اسلاید آمده است.

مبنای سنجش، مطالب گفته شده در کلاس است.

حضور در کلاس اجباری نیست ولی محتوای درس (برای تمرین ها و امتحانات) در کلاس توضیح داده می شود (همچنین ۲ نمره از نمره کل مربوط به کوییزهاست)

Topic	Reference
Recursion and Backtracking	Ch.1 and Ch.2 JeffE
Dynamic Programming	Ch.3 JeffE and Ch.15 CLRS
Greedy Algorithms	Ch.4 JeffE and Ch.16 CLRS
Amortized Analysis	Ch.17 CLRS
Elementary Graph algorithms	Ch.6 JeffE and Ch.22 CLRS
Minimum Spanning Trees	Ch.7 JeffE and Ch.23 CLRS
Single-Source Shortest Paths	Ch.8 JeffE and Ch.24 CLRS
All-Pairs Shortest Paths	Ch.9 JeffE and Ch.25 CLRS
Maximum Flow	Ch.10 JeffE and Ch.26 CLRS
String Matching	Ch.32 CLRS
NP-Completeness	Ch.12 JeffE and Ch.34 CLRS

برنامه نویسی پویا Dynamic Programming

- معمولا برای مسئله هایی استفاده می شود که چندین راه حل وجود دارد و می خواهیم یک راه حل با مقدار بهینه (کمترین یا بیشترین) محاسبه کنیم.

- "برنامه نویسی" با آنچه به عنوان مبانی برنامه نویسی خوانده اید ارتباطی ندارد. اینجا برنامه نویسی به خاطر استفاده از روش جدولی است.
  - مقدار راه حل های میانی معمولا در یک آرایه یا جدول ذخیره می شوند.

- یادگیری روش برنامه نویسی پویا از طریق مثال و حل کردن تمرین حاصل خواهد شد.

- به جای این که برای محاسبه جواب مسئله بزرگ، سراغ زیر مسئله های کوچکتر برویم، از زیر مسئله های کوچکتر شروع می کنیم به محاسبه جواب زیر مسئله های بزرگتر تا به مسئله بزرگ در ورودی برسیم.
  - جواب مسئله ای کوچک که برای حل چندین مسئله بزرگ تر لازم است، تنها یک بار محاسبه می شود
  - جواب مسئله بزرگ، با ترکیب جواب بهینه زیرمسئله های کوچکتر محاسبه می شود

محاسبه مقدار فیبوناچی

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$

- روش بازگشتی
  - زیرمسئله های تکراری
- محاسبه از پایین به بالا

Text Segmentation کلمه بندی متن

PRIMVSDIGNITASINTAMTENVISCIENTIANONPOTEST

ESSERESENIMSVNTPARVAEPROPEINSINGVLISLITTERIS

ATQVEINTERPVNCTIONIBUSVERBORVMOCCVPATAE

- We are given a string *A*[1 .. *n*] and a subroutine IsWord that determines whether a given string is a word (whatever that means), and we want to know whether *A* can be partitioned into a sequence of words.

- راه حل بازگشتی
- راه حل برنامه نویسی پویا

$$Splittable(i) = \begin{cases} \text{TRUE} & \text{if } i > n \\ \bigvee_{j=i}^{n} \left(IsWord(i, j) \wedge Splittable(j+1) \right) & \text{otherwise} \end{cases}$$

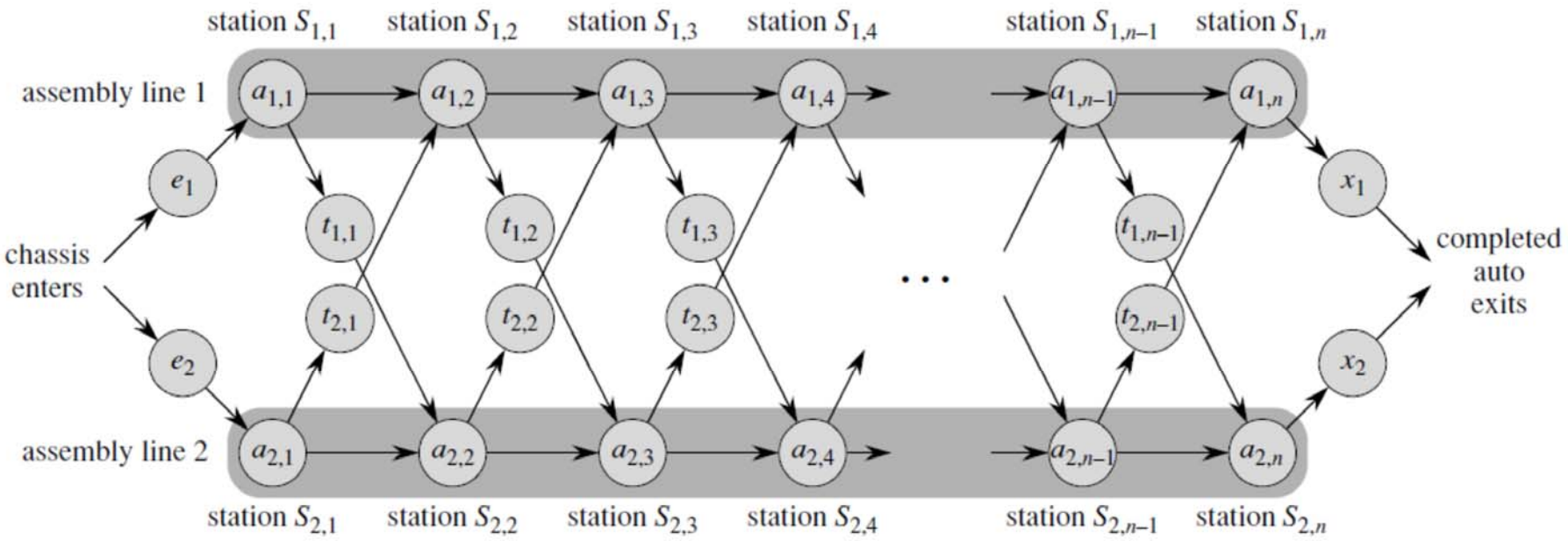
- *Splittable*(*i*) returns True if and only if the suffix *A*[*i* .. *n*] can be partitioned into a sequence of words
- *IsWord*(*i*, *j*) is shorthand for *IsWord*(*A*[*i* .. *j*])

```
FastSplittable(A[1..n]):
  SplitTable[n+1] ← True
  for i ← n down to 1
      SplitTable[i] ← False
      for j ← i to n
          if IsWord(i, j) and SplitTable[j+1]
              SplitTable[i] ← True
  return SplitTable[1]
```

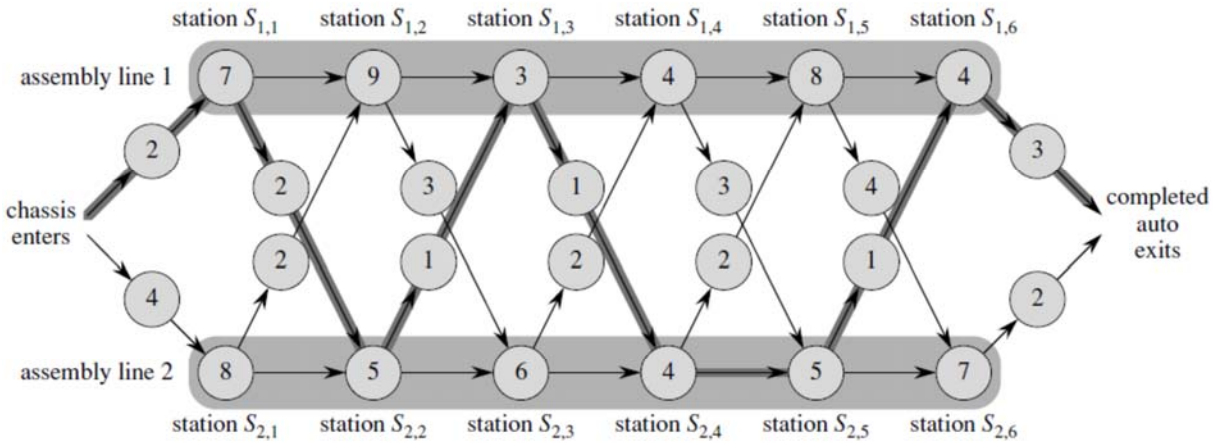
Memoization

- a variation of dynamic programming
  - maintains an entry in a table for the solution to each subproblem.
  - When the subproblem is first encountered during the execution of the recursive algorithm, its solution is computed and then stored in the table

Assembly-line scheduling زمانبندی خط تولید



مثال



(a)

j	1	2	3	4	5	6
$f_1[j]$	9	18	20	24	32	35
$f_2[j]$	12	16	22	25	30	37

$f^* = 38$

j	2	3	4	5	6
$l_1[j]$	1	2	1	1	2
$l_2[j]$	1	2	1	2	2

$l^* = 1$

(b)