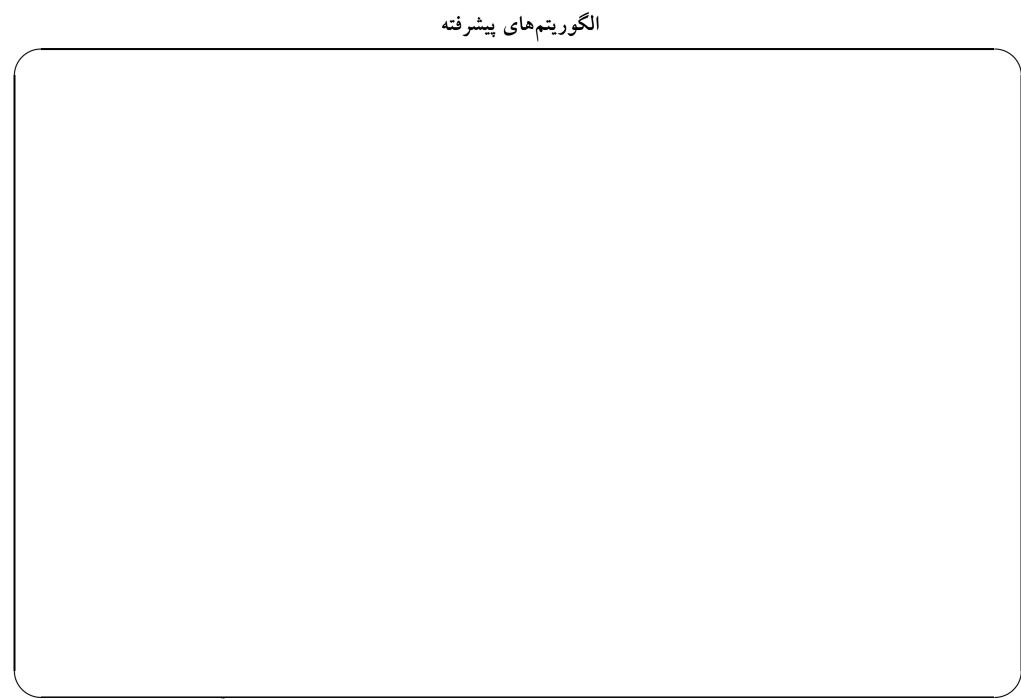
الگوریتمهای چندریسهای (Multithreaded Algorithms)

- الگوريتمهاي ترتيبي
- الگوريتمهاي موازي
- مدلهای چند پردازندهای
 - -- چندهستهاس
- -- مدل حافظهی مشترک
 - -- مدل توزیع شده



چند ریسهای

- ریسه (thread) چیست؟
- برنامه نویسی چندریسه ای ایستا معمولاً ریسه ها در طول اجرا وجود دارد
- برنامه نویسی چندریسه ای پویا (dynamic multithreaded programming) ریسه ها به صورت پویا تولید و حذف می شوند.

مدل چندریسهای پویا

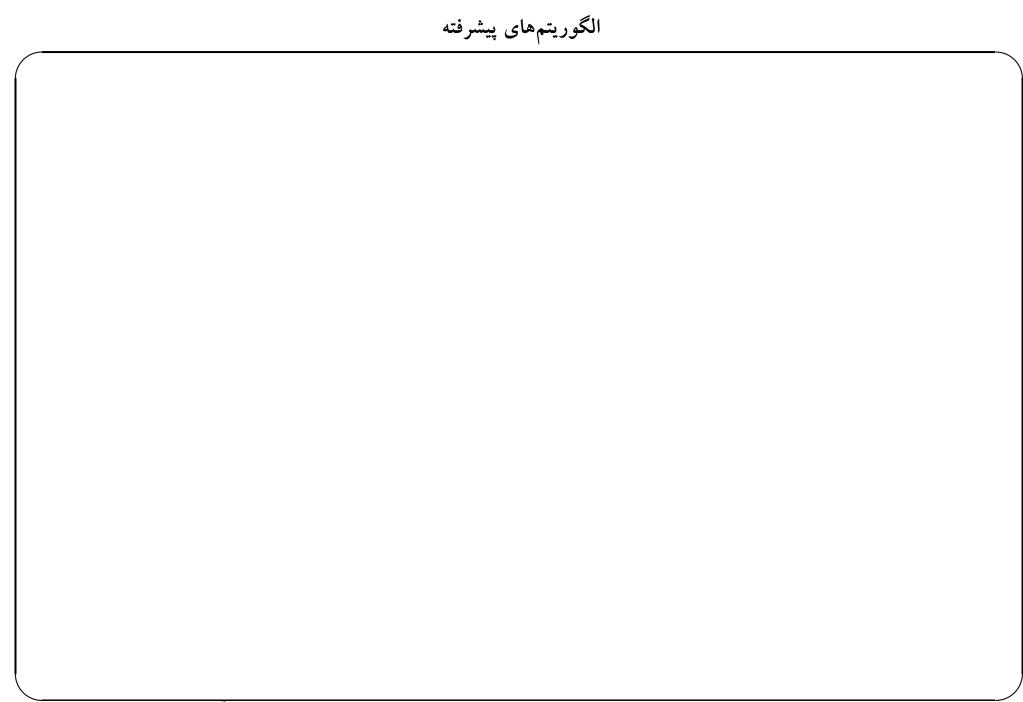
اعمال sync spwan و parallel

مثال: محاسبهی اعداد فیبوناچی

$$F_{\circ} = \circ$$

$$F_{1} = 1$$

$$F_{i} = F_{i-1} + F_{i-1}, i > 1$$



راه حل ترتیبی

- $\Phi = (1+\sqrt{\Delta})/$ ممل جمع دارد و از $\Theta(\Phi^n)$ است که $F_i 1$ عمل جمع دارد و
 - می توان در زمان خطی هم حل کرد.

• مىدانيم

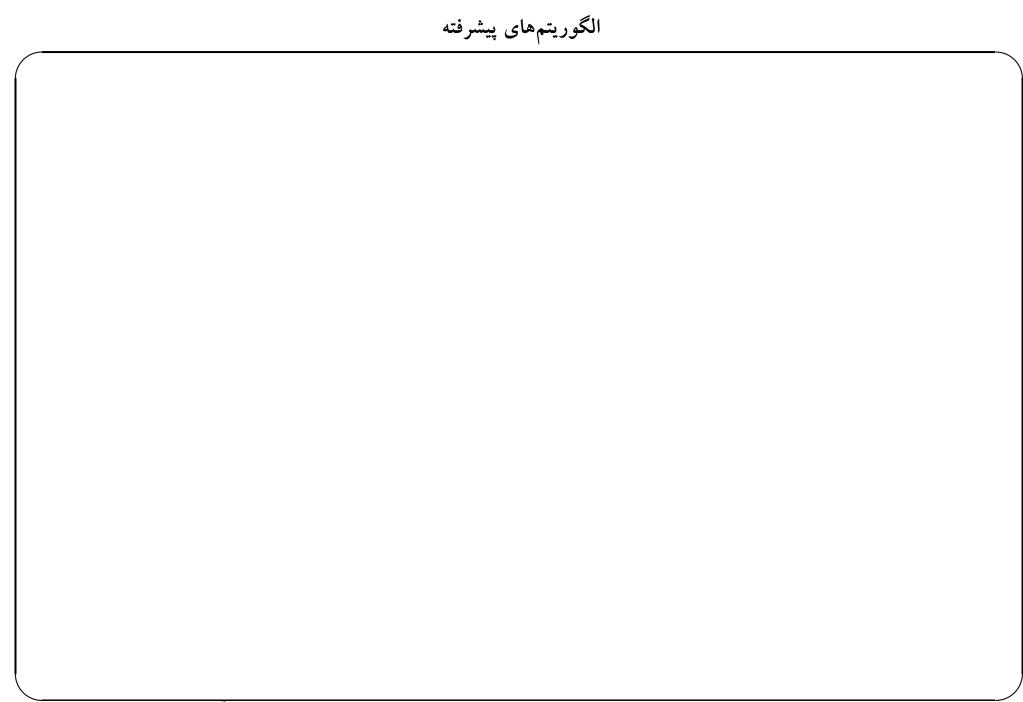
$$\begin{bmatrix} F_{n+}, & F_n \\ F_n & F_{n-} \end{bmatrix} = \begin{bmatrix} 1, & 1 \\ 1, & 0 \end{bmatrix}^n$$

برای $1 \geq n \geq n$ و طبق تعریف اصلی رابطه های زیر برقرارند.

$$\begin{bmatrix} F_{n+} & F_n \\ F_n & F_{n-} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-} \\ F_{n-} & F_{n-} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix}^{n-1} \cdot \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix}^n.$$

پس برای به دست آور دن F_n باید ماتریس $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ را n بار در خودش ضرب کنیم و این کار را می توان در $\Theta(\lg n)$ و بدون خطای محاسباتی دقیقاً محاسبه کرد.

راه حل چندریسهای





- strand، قطعه کدی که در دستورهای موازی ندارد
 - معیارها: span (زمان پایان) و work (کار)
 - مسير بحراني
 - T_P با P پردازنده، زمان اجرا

روشن است که

- (قانون كار) $T_P \geq T_1/P ullet$
- (قانون زمان پایان) $T_P \geq T_\infty$ •
- $T_1/T_P \leq P$ (speedup) تسریع
- اگر $T_1/T_P = \Theta(P)$ ، تسریع خطی
 - اگر $T_1/T_P = P$ ، تسریع کامل

- parallelism را موازات T_1/T_∞ •
- -- برابر میانگین کار که می تواند در هر مرحلهی مسیر بحرانی انجام شود.
 - -- کران بالای بیش ترین میزان تسریع ممکن با هر تعداد پردازنده
- -- حد بالای حالتی که ممکن است تسریع خطی داشت. اگر تعداد پردازنده ها از موازات بیش تر شود نمی توان انتظار تسریع خطی را داشت.

 $T_1/T_P \leq T_1/T_\infty < P$ یعنی اگر $P > T_1/T_\infty \leq P$ ، در آن صورت، $P > T_1/T_\infty \leq T_1/T_\infty$ یا اگر $P > T_1/T_\infty \leq P$ ، در آن صورت،

• در مورد اعداد فیبوناچی، $T_{\Lambda}=1$ و $\Lambda=T_{\infty}$ پس موازات برابر ۲.۱۲۵ $= \Lambda/\Lambda$ است.

- است. slackness می گوییم که عددی بین ۱ تا صفر است. $(T_1/T_\infty)/P$
 - $T_1/T_P < T_1/T_\infty < P$ در آنصورت، $\mathrm{slckness} < 1$ اگر ا
- -- اگر ا > slckness و به نزدیک بشود، در آن صورت، تسریع از خطی دور تر می شود.
 - -- اگر ۱ اگر ۱ slckness > ۱ تسریع به خطی نزدیک تر می شود.

زمانبندی

- زمانبندی برخط
- زمانبندی حریصانه
 - زمانبندی مرکزی
- $T_P \leq T_\infty$ و از طرفی $T_P = T_1/P$ ہیترین زمان پایان با P پردازندہ

قضیه: در یک کامپیوتر موازی ایده آل با زمان بندی حریصانه داریم

$$T_P \le T_1/P + T_\infty$$

اثبات: جمع تعداد مرحلههای کامل و ناکامل

 $\lfloor T_1/P \rfloor$:عداد مرحلههای کامل

 T_{∞} ابر است با کامل: حداکثر برابر است با

الگوريتمهاي پيشرفته

نتیجه: $T_P \leq TT_P^*$ در زمان بندی حریصانه بر روی کامپیوتر ایده آل موازی

$$T_P \le T_1/P + T_\infty$$

$$\le \Upsilon . \max\{T_1/P, T_\infty\}$$

$$\le \Upsilon T_P^*$$