

1.
(a)

算法: 输入 (问题描述)

输出 (答案)

正确性

确定性 (≥1个语义明确的基市操作)

可行性 (每个基市操作可实施, 常数时间)

△有穷性

有穷性: 又名(终止)问题 $\begin{cases} \{1\} & n \leq 1 \end{cases}$

* 递归 $Hailstone(n) = \begin{cases} \{n\} \cup Hailstone(n/2) & n \text{ 偶} \\ \{n\} \cup Hailstone(3n+1) & n \text{ 奇} \end{cases}$

$Hailstone(12) = \{12, 21, 64, 32, \dots, 1\}$

上下起伏, 不会一直上升, 但会一直下降. ^{有偶同数} 15个奇数 $3+1 \rightarrow 120$

* int hailstone(int n) { // 计算 Hailstone(n) 的长度

int length = 1;

while (1 < n) { (n%2) ? n = 3n+1 : n = n/2; length++; }

return length; // 返回长度

}

* 对于任何 n, $|Hailstone(n)| \text{ (长度)} < \infty$? 长度是否总是有限和长度? 不确定 (无终止, 也无终止)

* 程序中算法

(无循环, 栈溢)

好算法: 正确, 健壮, 可读

* 效率: 速度尽可能快; 存储空间尽可能少

Efficient Data Structure + Algorithm = Program

(b) 计算模型 DSA

性能测试 { 输入理想, 统一, 合理及的天度
互相造尺庭 以测量 DSA 的求解

问题规模

算法分析

{ 正确性

* 成本(新时间) + 所需存储空间

如何度量: 如何比较

(經濟學) 人國

：新華 (1)

(學) 經濟

外國文

(學) 中國文學 (1) 新華

(學) 中國文學 (2) 新華

外國文

121 113 101 91 81 71 61 51 41 31 21 11 1

(學) 中國文學 (3) 新華

(學) 中國文學 (4) 新華

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

(學) 中國文學 (5) 新華

121 113 101 91 81 71 61 51 41 31 21 11 1

(學) 中國文學 (6) 新華

121 113 101 91 81 71 61 51 41 31 21 11 1

(學) 中國文學 (7) 新華

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

(學) 中國文學 (8) 新華

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

121 113 101 91 81 71 61 51 41 31 21 11 1

* $T_A(P)$ = 算法 A 求解问题实例 P 的计算成本
意义不大，实例太多，如何归纳概括？

* 问题实例的规模，注意是决定计算成本的主要因素。

通常：规模缩小，成本接近
规模扩大，成本上升

· 特定算法 + 不同实例

令 $T_A(n)$ = 用算法 A 求解某一问题规模为 n 的实例所需的计算成本

依然不够，同一问题等规模的不同实例，也会有差异。

ex: n 个元素的最小生成树问题，可枚举 $C(n, 3)$ 种组合

* $T(n) = \max \{ T(P) \mid |P| = n \}$

在规模为 n 的所有实例中，找出最坏（成本最高）者

· 问题模型

特定问题 + 不同算法

为了解决问题，而需要抽象出一个问题模型 + 模型

· 图灵机 TM (Turing Machine)

* Tape: 依次均匀划分单元格

小格子 cell. 每个 cell 中 ~~只能~~ 没有一字符，默认为“#”

* Alphabet 字符有限

* Head (头, 读写头) 任何时刻，对准一个 cell。

· 每一指 向左 or 向右去下一个 cell, 可以 read/write cell 中字符。

* State TM 总是处于有限种状态中某一种

每个时刻，可按照规则，转入某一种 State

* Transition Function: $(q, c; d, L/R, p)$

→ 转换为 p 状态

“ q ” 状态为当前

当前 State

head 移动

当前 cell 中

向左

or 向右

在 cell 中写入

读入或修改

的字符

的字符

实例

年終時T: 07 (0.5) 總計 1.5 A 的 乘 = (9) T *

1. 計算總計 0.5, 0.5, 0.5, 0.5

2. 計算總計 0.5, 0.5, 0.5, 0.5

3. 計算總計 0.5, 0.5, 0.5, 0.5

4. 計算總計 0.5, 0.5, 0.5, 0.5

(0.5) T: 07 (0.5) 總計 1.5 A 的 乘 = (9) T *

年終時T: 07 (0.5) 總計 1.5 A 的 乘 = (9) T *

1. 計算總計 0.5, 0.5, 0.5, 0.5

2. 計算總計 0.5, 0.5, 0.5, 0.5

3. 計算總計 0.5, 0.5, 0.5, 0.5

4. 計算總計 0.5, 0.5, 0.5, 0.5

(0.5) T: 07 (0.5) 總計 1.5 A 的 乘 = (9) T *

1. 計算總計 0.5, 0.5, 0.5, 0.5

2. 計算總計 0.5, 0.5, 0.5, 0.5

(0.5) T: 07 (0.5) 總計 1.5 A 的 乘 = (9) T *

1. 計算總計 0.5, 0.5, 0.5, 0.5

2. 計算總計 0.5, 0.5, 0.5, 0.5

3. 計算總計 0.5, 0.5, 0.5, 0.5

4. 計算總計 0.5, 0.5, 0.5, 0.5

5. 計算總計 0.5, 0.5, 0.5, 0.5

6. 計算總計 0.5, 0.5, 0.5, 0.5

7. 計算總計 0.5, 0.5, 0.5, 0.5

8. 計算總計 0.5, 0.5, 0.5, 0.5

9. 計算總計 0.5, 0.5, 0.5, 0.5

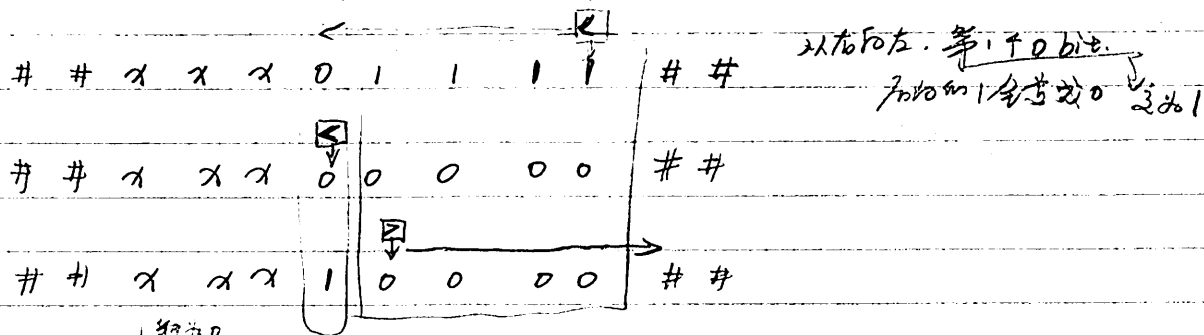
10. 計算總計 0.5, 0.5, 0.5, 0.5

11. 計算總計 0.5, 0.5, 0.5, 0.5

12. 計算總計 0.5, 0.5, 0.5, 0.5

(0.5) T: 07 (0.5) 總計 1.5 A 的 乘 = (9) T *

• TM实例: Increase: 递增: 将二进制非负整数加1



($\leftarrow, 1, 0, L, \leftarrow$) // 左移, $1 \rightarrow 0 \Rightarrow$ 前移4次

($\leftarrow, 0, 1, R, \rightarrow$) // $0 \rightarrow 1$ $\leftarrow \Rightarrow \rightarrow$ 掉头. (遇到0, 改为1. 掉头)

($\leftarrow, \#, 1, R, \rightarrow$) // 找到# 进一位

($\rightarrow, 0, 0, R, \rightarrow$) // 右移

($\rightarrow, \#, \#, L, h$) // 复位, 停机

规范 ~ 接口

• RAM模型: Random Access machine. (STM均有无限空间)

* Register 顺序编号. 总数为有限 $R[0], R[1], R[2], \dots$

• 每步操作只需常数时间

$R[i] \leftarrow c$

$R[i] \leftarrow R[j]$ 将 $R[j]$ 中值赋给 $R[i]$, 直接取址

$R[i] \leftarrow R[R[j]]$ 间接取址

$R[R[i]] \leftarrow R[j]$

// 循环及子程序本身非常数操作

$R[i] \leftarrow R[i] \pm R[k]$

IF $R[i] = 0$ GOTO t .

GOTO t

STOP

\Rightarrow 在这些操作中, 将算法运行时间 \propto 算法所需执行的 基本操作数

• RAM实例: Floor 向下取整的除法, $0 \leq c, 0 < d$

$$\lfloor c/d \rfloor = \max \{ x \mid d \cdot x \leq c \}$$

$$= \max \{ x \mid d \cdot x < 1 + c \}$$

算法: 反推从 $R[0] = 1 + c$ 中减去 $R[1] = d$, 记下 减法 所需执行的次数

1. 证明：若 a, b, c 是正实数，则 $a^2 + b^2 + c^2 \geq ab + bc + ca$ 。

证：由 $(a-b)^2 \geq 0$ 得 $a^2 + b^2 \geq 2ab$ 。

同理，由 $(b-c)^2 \geq 0$ 得 $b^2 + c^2 \geq 2bc$ 。

由 $(c-a)^2 \geq 0$ 得 $c^2 + a^2 \geq 2ca$ 。

将以上三式相加，得 $2(a^2 + b^2 + c^2) \geq 2(ab + bc + ca)$ 。

即 $a^2 + b^2 + c^2 \geq ab + bc + ca$ 。

证毕。

2. 证明：若 a, b, c 是正实数，则 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 3$ 。

证：由 $\frac{a}{b} + \frac{b}{a} \geq 2$ 得 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2 + \frac{c}{a}$ 。

同理，由 $\frac{b}{c} + \frac{c}{b} \geq 2$ 得 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2 + \frac{a}{b}$ 。

将以上两式相加，得 $2(\frac{a}{b} + \frac{b}{c} + \frac{c}{a}) \geq 4 + \frac{a}{b} + \frac{b}{c}$ 。

即 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2$ 。

同理，由 $\frac{c}{a} + \frac{a}{c} \geq 2$ 得 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2 + \frac{c}{a}$ 。

将以上两式相加，得 $2(\frac{a}{b} + \frac{b}{c} + \frac{c}{a}) \geq 4 + \frac{a}{b} + \frac{b}{c}$ 。

即 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2$ 。

将以上三式相加，得 $3(\frac{a}{b} + \frac{b}{c} + \frac{c}{a}) \geq 6$ 。

即 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2$ 。

证毕。

证毕。

3. 证明：若 a, b, c 是正实数，则 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 3$ 。

证：由 $\frac{a}{b} + \frac{b}{a} \geq 2$ 得 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2 + \frac{c}{a}$ 。

同理，由 $\frac{b}{c} + \frac{c}{b} \geq 2$ 得 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2 + \frac{a}{b}$ 。

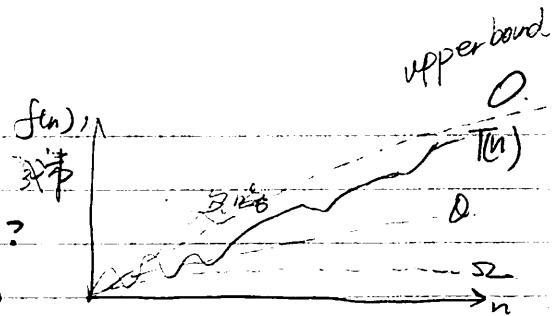
将以上两式相加，得 $2(\frac{a}{b} + \frac{b}{c} + \frac{c}{a}) \geq 4 + \frac{a}{b} + \frac{b}{c}$ 。

即 $\frac{a}{b} + \frac{b}{c} + \frac{c}{a} \geq 2$ 。

(c) 大O记号

渐进分析: 随着问题规模的 n , 成本如何个?
 渐近分析: 关心 是否大, 成本 增长趋势

Asymptotic analysis: 当 $n \gg 2/\epsilon$, 对于 n 足够大, 成本增长趋势 $T(n) = ?$



大O记号 (big-O notation)

$T(n) = O(f(n))$ 当 $\exists c > 0$, 当 $n \gg 2/\epsilon$ 后, 有 $T(n) \leq c \cdot f(n)$

ex:
$$\sqrt{5n - [3n \cdot (n+2) + 4] + 6} < \sqrt{5n - [6n^2 + 4] + 6} < \sqrt{5n - 6n^2 + 2} < \sqrt{5n} < \sqrt{5} \sqrt{n} = O(\sqrt{n})$$

* 局部: 忽略反映趋势

* 常数忽略: $O(f(n)) = O(c \cdot f(n))$

* 低次项忽略: $O(n^a + n^b) = O(n^a)$, $a > b > 0$

其它记号:

下界 (lower bound)

* $T(n) = \Omega(f(n))$: $\exists c > 0$, 当 $n \gg 2/\epsilon$ 后, 有 $T(n) \geq c \cdot f(n)$

* $T(n) = \Theta(f(n))$: $\exists c_1 > c_2 > 0$ 当 $n \gg 2/\epsilon$ 后, 有 $c_1 f(n) \leq T(n) \leq c_2 f(n)$

大O记号

$O(1)$:

* 常数 (Constant function)

$2 = 2013 = 2013 \times 2013 = O(1)$, 甚至 $2013^{2013} = O(1)$

* 效率最高的算法: ① 不含循环 (递归, 调用, 递归) 的/3/5/3/6

② 不建不创临时: for ($i=0$; $i < n$; $i += n/2013 + 1$).

for ($i=1$; $i < n$; $i = \lfloor i/2 \rfloor$); // $\log n$ 个步骤

$\forall c > 0, \log n = O(n^c)$

按效率 $\rightarrow O(1)$

③

④ 除或取模 (递归) 若 $(2 \div (n \div n)) \% 0 = O(n)$;

$O(\log n)$: 对数 $O(\log n)$ $\log n \mid \lg n \mid \log_{10} n \mid \log_{2013} n$

* 底数常数, 无所谓

$\forall a, b > 0, \log_a n = (\log_b a) \cdot \log_b n = O(\log n)$

* 常数及幂, 无所谓

$\forall c > 0, \log n^c = c \log n = \Theta(\log n)$

* 对数多倍: $1.23 \cdot \log^2 n + \log_{10}(n^2 - n + 1) = O(\log^2 n)$

[illegible]

Contract C - old, 2002

陳述之與陳述之

1941-1942

[illegible]

~~SECRET~~

~~$$(100) = \frac{2 \times 10^4}{100} = 200, (100) = 200 \times 100 = 20000 = 2$$~~

(1) + 600(112+8; 112个 : 125) - 10% : 112个不整不②

(1) 3-2-10-5

10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125. 126. 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139. 140. 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153. 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181. 182. 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195. 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209. 210. 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223. 224. 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237. 238. 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251. 252. 253. 254. 255. 256. 257. 258. 259. 260. 261. 262. 263. 264. 265. 266. 267. 268. 269. 270. 271. 272. 273. 274. 275. 276. 277. 278. 279. 280. 281. 282. 283. 284. 285. 286. 287. 288. 289. 290. 291. 292. 293. 294. 295. 296. 297. 298. 299. 300. 301. 302. 303. 304. 305. 306. 307. 308. 309. 310. 311. 312. 313. 314. 315. 316. 317. 318. 319. 320. 321. 322. 323. 324. 325. 326. 327. 328. 329. 330. 331. 332. 333. 334. 335. 336. 337. 338. 339. 340. 341. 342. 343. 344. 345. 346. 347. 348. 349. 350. 351. 352. 353. 354. 355. 356. 357. 358. 359. 360. 361. 362. 363. 364. 365. 366. 367. 368. 369. 370. 371. 372. 373. 374. 375. 376. 377. 378. 379. 380. 381. 382. 383. 384. 385. 386. 387. 388. 389. 390. 391. 392. 393. 394. 395. 396. 397. 398. 399. 400. 401. 402. 403. 404. 405. 406. 407. 408. 409. 410. 411. 412. 413. 414. 415. 416. 417. 418. 419. 420. 421. 422. 423. 424. 425. 426. 427. 428. 429. 430. 431. 432. 433. 434. 435. 436. 437. 438. 439. 440. 441. 442. 443. 444. 445. 446. 447. 448. 449. 450. 451. 452. 453. 454. 455. 456. 457. 458. 459. 460. 461. 462. 463. 464. 465. 466. 467. 468. 469. 470. 471. 472. 473. 474. 475. 476. 477. 478. 479. 480. 481. 482. 483. 484. 485. 486. 487. 488. 489. 490. 491. 492. 493. 494. 495. 496. 497. 498. 499. 500. 501. 502. 503. 504. 505. 506. 507. 508. 509. 510. 511. 512. 513. 514. 515. 516. 517. 518. 519. 520. 521. 522. 523. 524. 525. 526. 527. 528. 529. 530. 531. 532. 533. 534. 535. 536. 537. 538. 539. 540. 541. 542. 543. 544. 545. 546. 547. 548. 549. 550. 551. 552. 553. 554. 555. 556. 557. 558. 559. 560. 561. 562. 563. 564. 565. 566. 567. 568. 569. 570. 571. 572. 573. 574. 575. 576. 577. 578. 579. 580. 581. 582. 583. 584. 585. 586. 587. 588. 589. 590. 591. 592. 593. 594. 595. 596. 597. 598. 599. 600. 601. 602. 603. 604. 605. 606. 607. 608. 609. 610. 611. 612. 613. 614. 615. 616. 617. 618. 619. 620. 621. 622. 623. 624. 625. 626. 627. 628. 629. 630. 631. 632. 633. 634. 635. 636. 637. 638. 639. 640. 641. 642. 643. 644. 645. 646. 647. 648. 649. 650. 651. 652. 653. 654. 655. 656. 657. 658. 659. 660. 661. 662. 663. 664. 665. 666. 667. 668. 669. 670. 671. 672. 673. 674. 675. 676. 677. 678. 679. 680. 681. 682. 683. 684. 685. 686. 687. 688. 689. 690. 691. 692. 693. 694. 695. 696. 697. 698. 699. 700. 701. 702. 703. 704. 705. 706. 707. 708. 709. 710. 711. 712. 713. 714. 715. 716. 717. 718. 719. 720. 721. 722. 723. 724. 725. 726. 727. 728. 729. 730. 731. 732. 733. 734. 735. 736. 737. 738. 739. 740. 741. 742. 743. 744. 745. 746. 747. 748. 749. 750. 751. 752. 753. 754. 755. 756. 757. 758. 759. 760. 761. 762. 763. 764. 765. 766. 767. 768. 769. 770. 771. 772. 773. 774. 775. 776. 777. 778. 779. 780. 781. 782. 783. 784. 785. 786. 787. 788. 789. 790. 791. 792. 793. 794. 795. 796. 797. 798. 799. 800. 801. 802. 803. 804. 805. 806. 807. 808. 809. 810. 811. 812. 813. 814. 815. 816. 817. 818. 819. 820. 821. 822. 823. 824. 825. 826. 827. 828. 829. 830. 831. 832. 833. 834. 835. 836. 837. 838. 839. 840. 841. 842. 843. 844. 845. 846.

$$P = \frac{1}{2} \left(\frac{1}{2} + \frac{1}{2} \right) = \frac{1}{2}$$
$$f(x) = \frac{1}{2} + \frac{1}{2} \cos(2x) = \frac{1}{2} + \frac{1}{2} (\cos^2(x) - \sin^2(x)) = \frac{1}{2} + \frac{1}{2} (\cos^2(x) - 1 + \cos^2(x)) = \cos^2(x)$$

* $O(n^c)$ 多项式 有穷时间

$O(n)$... 线性时间

$O(n^2)$... 平方时间

linear function: $O(n)$

* 指数 $O(2^n)$

从 $O(n^c)$ 到 $O(2^n)$ 是指数级增长

指数 $T(n) = a^n$

$\forall c > 1, n^c = O(2^n)$

$n^{1000} = O(1.000001^n) = O(2^n)$

$1.000001^n = \Omega(n^{1000})$

• 2-SUBSET

→ is NP-complete 不存在可以在多项式时间内解决的问题

(d) 算法分析

正确性 (正确性 × 效率)

复杂度: { 迭代: 级数求和
递归: 递归跟踪 + 递归树
递归 + 记忆

收敛的, 但有限

调和级数: $1 + 1/2 + 1/3 + \dots + 1/n = O(\log n)$
对数级数: $\log 1 + \log 2 + \log 3 + \dots + \log n = \log(n!) = O(n \log n)$

(1) 级数

* 算术级数: 求和平方级数

$$T(n) = 1 + 2 + \dots + n = n(n+1)/2 = O(n^2)$$

* 平方级数: 以幂级数高一阶 $\sum_{k=1}^n k^d \approx \int_0^n x^d dx = O(n^{d+1})$

$$T_2(n) = 1^2 + 2^2 + \dots + n^2 = n(n+1)(2n+1)/6 = O(n^3)$$

$$T_4(n) = 1^4 + 2^4 + \dots + n^4 = n(n+1)(2n+1)(3n^2+3n-1)/30 = O(n^5)$$

* 几何级数 ($a > 1$): 求和项级数

$$T_a(n) = a^0 + a^1 + \dots + a^n = (a^{n+1} - 1)/(a - 1) = O(a^n)$$

ex: $a=2, 1 + 2^1 + \dots + 2^n = O(2^{n+1})$

* 收敛级数

$$1/1 + 1/2 + 1/3 + 1/4 + \dots + 1/(n+1) - 1/n = O(1/n)$$

$$1/2^2 + \dots + 1/n^2 < 1 + 1/2^2 + \dots = \pi^2/6 = O(1)$$

$$1/3 + 1/7 + 1/8 + 1/15 + \dots + 1/35 = 1 = O(1)$$

1. 100% (100%) *
 2. 100% ... (100%)
 3. 100% ... (100%)
 (100%) : 100% ...

1. 100% (100%) (100%)

(100%) (100%) *

100% (100%)

(100%) = 100%

(100%) = (100% + 100%) = 100%

(100%) = 100%

100% - 100%

1. 100% (100%) (100%)

100% (100%)

(100%) (100%)

100% (100%)

100% (100%)

100% (100%)

100% (100%)
 100% (100%)
 100% (100%)
 100% (100%)

100% (100%)

(100%) = 100%

(100%) = 100%

(100%) = 100%

(100%) = 100%

100% (100%)

(100%) = 100%

(100%) = 100%

100% (100%)

100% (100%)

(100%) = 100%

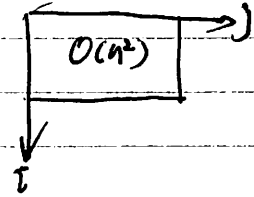
(100%) = 100%

$$1 \quad 2^2 \quad 3^2 \quad 4^2 \quad 5^2 \dots n^2$$

(2) 循环:

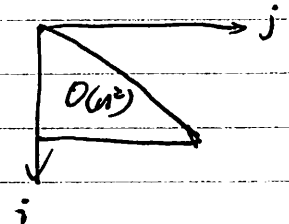
① for (int i=0; i<n; i++)
for (int j=0; j<n; j++)
operation(i,j);

计算结果: $\sum_{i=0}^{n-1} n = n + n + n + \dots + n = n \times n = O(n^2)$



② for (int i=0; i<n; i++)
for (int j=0; j<i; j++)
operation(i,j);

计算结果: $\sum_{i=0}^{n-1} i = 0 + 1 + \dots + (n-1) = \frac{n(n-1)}{2} = O(n^2)$



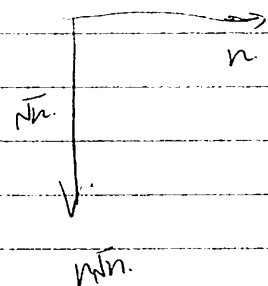
③ for (int i=0; i<n; i++)
for (int j=0; j<i; j+=2013)
operation(i,j);



右轴刻度为 2013

④ for (int i=1; i<n; i*=2)
for (int j=0; j<i; j++)
operation(i,j);

左轴刻度 $i = i \times 2$

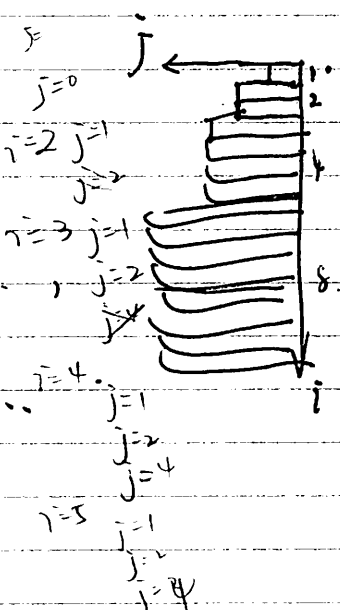


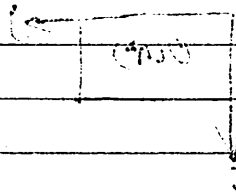
计算结果: $1 + 2 + 4 + \dots + 2^{\lfloor \log_2(n-1) \rfloor} = O(n)$

⑤ for (int i=0; i<n; i++)
for (int j=i; j<i; j+=j)
operation(i,j);

$i = 0, 1, 2, 3 \sim 4, 5 \sim 8, 9 \sim 16 \dots$
 $= 0 + 0 + 1 + 2 \times 2 + 3 \times 4 + 8 \times 8 + 5 \times 16 + \dots$

$= O(\log n \times 2^{\log n})$



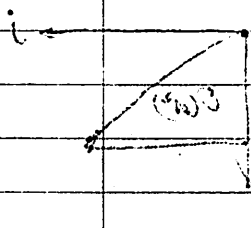


(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100)

(100 : 100 : 100 : 100) 100

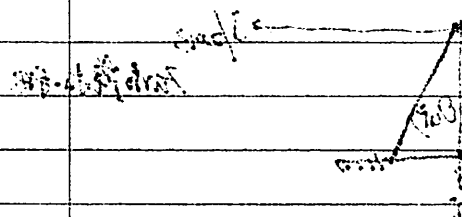


(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100)

(100 : 100 : 100 : 100) 100



(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100)

(100 : 100 : 100 : 100)

(100 : 100 : 100 : 100) 100

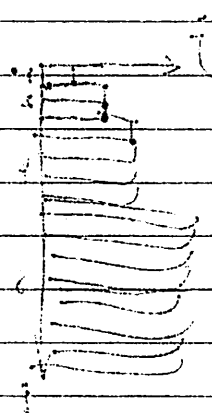
(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100)

(100 : 100 : 100 : 100)

(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100)



(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100)

(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100) 100

(100 : 100 : 100 : 100)

(100 : 100 : 100 : 100) 100

fib(n) 递归

$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$$

递归 $T(n) = T(n-1) + T(n-2) + 1; \quad T(0) = T(1) = 1 \quad n > 1$

令 $S(n) = [T(n) + 1] / 2$

则 $S(0) = [T(0) + 1] / 2 = 1, \quad S(1) = 1 = \text{fib}(2)$
 $= \text{fib}(1)$

则 $S(n) = S(n-1) + S(n-2) = \text{fib}(n+1)$

$$T(n) = 2 \times S(n) - 1 = 2 \times \text{fib}(n+1) - 1 = O(\text{fib}(n+1))$$

$$E\left[\frac{d}{dt}L(n)\right]dt$$

$$(1-n)dt = (1-n)dt + (1-n)dt$$

$$1 < n \quad 1 = (1-n)T = (1-n)T \cdot 1 + (1-n)T \cdot 1 + (1-n)T \cdot 1$$

$$E[1 + (1-n)T] = (1-n)T$$

$$(1-n)dt = 1 - (1-n)T \quad 1 = E[1 + (1-n)T] = (1-n)T$$

$$(1-n)dt =$$

$$(1-n)dt = (1-n)T + (1-n)T = (1-n)T$$

$$((1-n)dt) \cdot 0 = 1 - (1-n)dt \cdot 1 = 1 - (1-n)T = (1-n)T$$

向量 - 元组

1. `template < typename T > Vector { ... },`

`Vector < int > myVector;`
`float`
`char`

`Vector < BinTree > forest;`

2. 随机访问 (元素访问)

• ADT: `V.get(r)` `V.put(r, e)` 读、写

• 不如数组中 `A[r]` 便捷

• 可免 "overload" `[]`

`template < typename T > // 0 ≤ r < _size`

`T & Vector<T>::operator[] (Rank r) constant const { return _elem[r]; }`

• 记号: 对外的 `V[r]` 对内的 `V._elem[r]`

右值 $T x \leftarrow V[r] + U[s] * W[x]$

左值: `V[r] ← (T) (2Ax + 3);`

3. 插入:

`template < typename T > // e 作为秩为 r 的元素插入, 0 ≤ r < _size.`

`Rank Vector<T>::insert (Rank r, T const & e) {`

`expand(); // if necessary`

`for (int i = _size; i > r; i--) // 自后向前`

`_elem[i] = _elem[i-1]; // 从后向前, 每个元素右移一位`

`_elem[r] = e; _size++; // 为 e 于 r 位置, size+1`

`return r;`

`}`

① for 循环, r 右边所有元素右移一位, 顺序是从后向前 (即从右向左移动)

② `expand()` 防止越界

4. 区间删除 (左移)

`template < typename T > // 删除区间 [lo, hi), 0 ≤ lo ≤ hi ≤ size.`

`int Vector<T>::remove (Rank lo, Rank hi) { // O(n-hi)`

`if (lo == hi) return 0; // 空区间, 单独考虑此种情况`


```

while (hi < _size) _elem[lo++] = _elem[hi++]; // [hi, _size) 元素移到 [lo, hi) 位置
size = lo; shrink(); // 更新规模。若有必要则删除多余元素。
return hi - lo; // 返回新数组的末尾

```

5. 单元素删除

```

[r] = [r, r+1)
template <typename T> // 删除下标为 r 的元素。0 <= r < size.
T Vector<T>::remove (Rank r) { // O(n-r)
    T e = _elem[r] // 备份被删除元素
    remove(r, r+1) // 调用区间删除算法
    return e;
}

```

反过来，区间删除调用单元素删除如何？效率更低！
 $O(n^2)$

6. 查找

- 无符号：T 为可判等的算术类型，或已重载 "==" "!="
- 有符号：T 可以放 - - - - - "<" 或 ">"

```

template <typename T> // 0 <= lo < hi < size
Rank Vector<T>::find (T const &e, Rank lo, Rank hi) const {
    while (lo < hi--) { // 逆序查找 (右->左)
        if (e == _elem[hi]) return hi; // hi < lo 则表示未找到，返回 hi 为命中点
    }
}

```

最好情况 $O(1)$ 最坏为 $O(n)$ \rightarrow input-sensitive

7. 唯一化：算法 (删除重复的)

```

int Vector<T>::deduplicate () { // 删除重复元素
    int oldSize = _size; // 记录原始大小
    Rank i = 1; // 从 _elem[1] 开始
    O(n) while (i < _size) // 向前遍历，检查每个元素 _elem[i]
        if (find(_elem[i], 0, i) < 0) { // 元素 i 之前未出现过
            i++; // 元素 i 是唯一的
        } else {
            remove(i); // 否则删除重复元素
        }
    return oldSize - _size;
}

```

复杂度分析：
 唯一化：算法 (删除重复的)
 复杂度：
 最好情况 $O(1)$ 最坏为 $O(n)$ \rightarrow input-sensitive

Diagram: A horizontal bar representing an array. The left part is labeled 'prefix' and contains 'x'. The right part is labeled 'suffix' and contains 'x' and 'y'. Above the bar, 'prefix' and 'suffix' are written. Below the bar, 'find(x)' and 'find(y)' are written. The bar is divided into three sections: 'prefix', 'x', and 'suffix'.

8. 遍历 (visit 对各个元素分别实施)

1. 通过指针, 完成或局部修改

```
template <typename T>
```

```
void Vector<T>::traverse (void (*visit)(T&))
```

```
{ for (int i=0; i<_size; i++) visit (_elem[i]); }
```

更通用 \Rightarrow 函数对象, 局部修改

```
template <typename T> template <typename VST>
```

```
void Vector<T>::traverse (VST& visit)
```

```
{ for (int i=0; i<_size; i++) visit (_elem[i]); }
```

实例: 统一将向量中所有元素加1.

首先, 实现一个可使单个元素加1的类

```
template <typename T> // 假设 T 有重载 +1 或已 overload "++"
```

```
struct Increase { 函数对象: 通过 overload "()" 实现
```

```
virtual void operator()(T& e) { e++; }
```

```
};
```

使用:

```
template <typename T> void increase (Vector<T> & V) {
```

```
    V.traverse (Increase<T>());
```

```
}
```

(संस्कृत-विशेषज्ञ) श्री .

一、國書
二、圖書

→ $\text{C}_2\text{H}_5\text{COOH} \rightarrow \text{C}_2\text{H}_5\text{COCl}$

(CRT) (4-18-70) (JAN) 22-10-1960

1 (Trombato) 25th (1951) 200-201 (1951) 202

國史之爲書錄其始也

$\langle T_{\mu\nu} \rangle$ = $\langle T_{\mu\nu} \rangle$ = $\langle T_{\mu\nu} \rangle$

(step 2) $\frac{1}{\sqrt{2}}$

2. (17 months) until 6.7.71; 952-07 (009 501)-01 3

[illegible]

陳子昂集卷之四

[illegible]

34.5" W longitude (E): 117.545; constant 1000

Letter (S & T) to the Editor

13

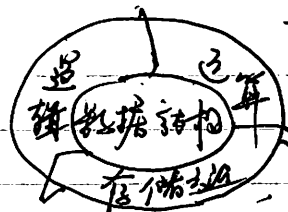
100

Temperature - dependent (Vapor > T > 81 K)

1. (b) \rightarrow $\frac{1}{2} \log \left(\frac{1}{2} \right)$

3.

数据结构: 实体 + 关系



输入 → 运算 → 输出

逻辑结构: 逻辑关系组织起来的逻辑

线性: 线性表 (stack, queue, table, 串)

非线性: tree (二叉树) (图及图)

图: 有向图, 无向图 (受限, 最通用)

图 = 树 = 二叉树 = 线性表

存储结构: 逻辑 → physical memory 的映射

顺序 (非线性), 链表 (非线性), 索引 (非线性), 散列 (非线性)

抽象数据类型 ADT (Abstract data type)

定义一组公共的数据模型 (接口对象)

<数据对象 D, 数据操作 P>

先定义逻辑结构 (数据对象及关系)

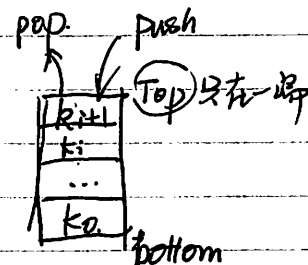
再运算: 数据操作

ex. Stack 的 ADT

逻辑结构: 线性表

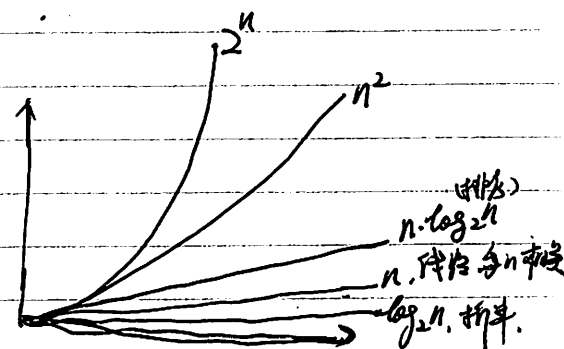
操作特点: 限制为内端口 < 只允许一端输入, 删除

push, pop, top, isEmpty, 写读



template <class T> 模板类

```
class Stack {
    void clear();
    bool push(const T item);
    bool pop(T & item);
    bool top(T & item);
    bool isEmpty();
    bool isFull;
}
```



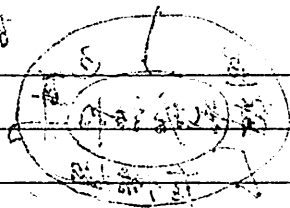
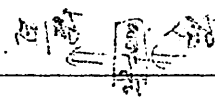
大O表示法的运算规则

加法规则 $f_1(n) + f_2(n) = O(\max(f_1(n), f_2(n)))$

例: if, switch

乘法: $f_1(n) * f_2(n) = O(f_1(n) * f_2(n))$

for, while, do-while, 结构



input + output = output

input + output = output
(input, output, output) output: output

(input, output, output) output: output

(input, output, output) output: output

input + output = output

input + output = output: output

input + output = output

(input, output, output) output: output

(input, output, output) output: output

(input, output, output) output: output

input + output = output

(input, output, output) output: output

input

output: output

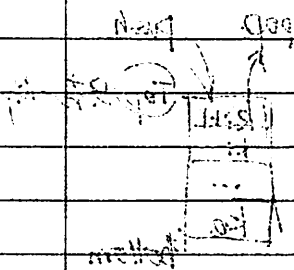
input + output = output

input + output = output

input + output = output

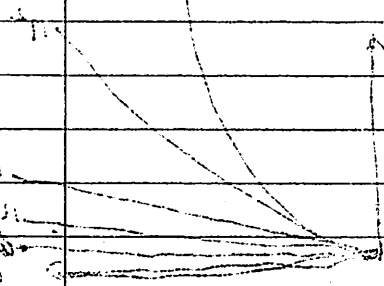
input + output = output

input + output = output



input

input



input

input

input

input + output = output

input + output = output

input + output = output

input + output = output

input + output = output

input + output = output

input + output = output

input + output = output

input + output = output

(input, output, output) output: output

input + output = output

(input, output, output) output: output

input + output = output

线性表 $\{a_0, a_1, \dots, a_n\}$ 0个或多个元素组成的有穷序列
 特点: 顺序存储 索引/下标, 长度 n , 长度为 (record, 数据项)
 特点: 小范围, 0则为空表

二元组 $B = (K, R)$ $K = \{a_0, a_1, \dots, a_{n-1}\}$ $R = \{r\}$

前驱后继关系, 反对称性
 传递性

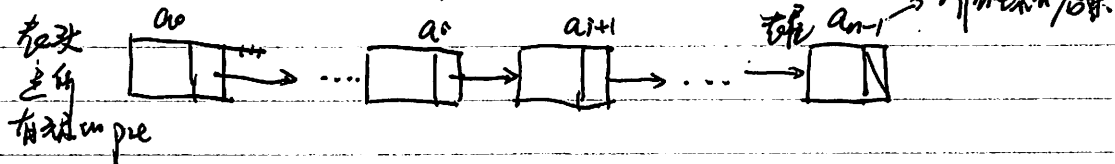
唯一-start node. no pre, 只有一个直接前驱

$\langle a_i, a_j \rangle \rightarrow \langle a_j, a_k \rangle$
 $\langle a_i, a_k \rangle$

唯一-end node. no 后继.

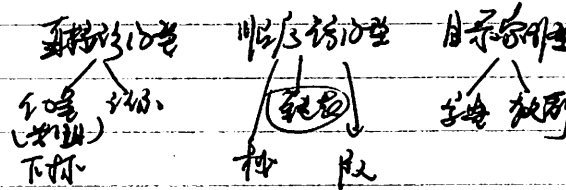
内部结点: 1 pre, 1 后继.

$\langle a_i, a_{i+1} \rangle$, a_i 是 a_{i+1} pre, a_{i+1} 是 a_i 后继.



特点: 均匀性 (同一数据类型和长度)
 顺序性

ex: 线性表, 栈, 队列, 数组 (索引+地址)
 广义表, 多维数组, 文件



线性表入栈操作
 stack LIFO: 插入删除在一端
 Queue FIFO: 插入左端, 删除另一端

逻辑结构

逻辑 attribute:
 { 长度
 head
 tail
 current position

存储结构 < 线性表 物理关系 > 逻辑关系

链式表: 单 (指针) 双向 (指针)
 循环 (指针)

运算: 建立, 删除, 增删, 查, 改, 排序, 检索

template <class T> class List {

void clear();
 bool isEmpty();
 bool append (const T value); // 表尾加一个
 bool insert (const int p, const T value);
 bool delete (const int p);

bool getPos (int p, const T value);
 bool getValue (const int p, T value);
 bool setValue (const int p, const T value);

張永年 張永年

15/10/20

$$L_2 = \emptyset \quad L_1 \cap L_2 = \emptyset \quad (S, \tau) = S \text{ ist MC}$$

مجلسه ۱۳۴۵

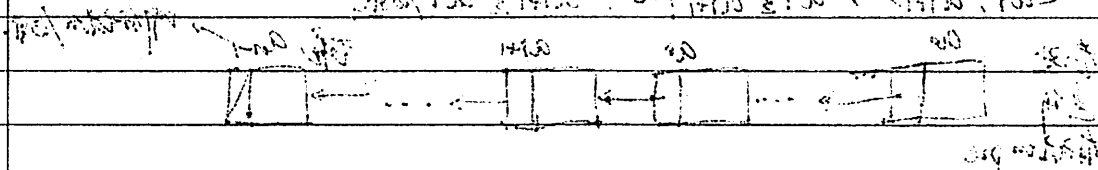
455, 1264

















[illegible]

April 11, 1944 - 87

第 1 页

1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 26



| | | | | | | | |
|--|--|--|--|--|--|--|--|
| <p>   </p> | <p>   </p> | <p>   </p> | <p>   </p> | <p>   </p> | <p>   </p> | <p>   </p> | <p>   </p> |
|--|--|--|--|--|--|--|--|

(此項試驗係第一回) 於 2011. 12. 24
於 2011. 12. 24

$$\frac{1}{2}(\frac{1}{2} + \frac{1}{2}) = \frac{1}{2}$$

新學人必知

2000 1110 第 1110 号

中二部 中二部 中二部 中二部 中二部

24 51 27 28

2017-2018

Y. A. S. N.

—بعضی

nothing (1/20/50)

1960年

陳永發 (1942) 李國章

4-2-11

22
1941

[illegible]

39-615

(Singer) and, 2. and 3. top left

1. What is the purpose of the experiment?

| | | | |
|--------|------------|----------|----------|
| Cal ST | april 1961 | 10/11/61 | 10/11/61 |
|--------|------------|----------|----------|

[illegible]

(1) पृष्ठ संख्या

[illegible][illegible]

(extra time)

~~(1) (2) (3) (4) (5) (6) (7) (8) (9) (10)~~

(1994) 1994

顺序表 (SeqList): 定长的一维数组存储结构

元素同类型 (所有元素类型) 逻辑

随机访问: 一个元素一个索引值 (下标) 指定位置 随机存储

$$Loc(k_i) = Loc(k_0) + \underbrace{c \times i}_{\text{sizeof(element)}}$$

```
class SeqList: public List<T> {
```

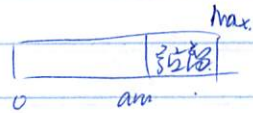
private:

T * aList; // 连续内存.

int maxSize; // 最大长度

int curLen; // 当前长度

int position;



public:

```
SeqList(const int size) {
```

```
    maxSize = size; aList = new T[maxSize];
```

```
    curLen = position = 0;
```

```
}
```

```
~SeqList() {
```

```
    delete[] aList;
```

```
}
```

1. Introduction and Background

2. Methodology and Results

3. Discussion and Conclusion

4. References and Appendix

5. Summary

6. Future Work and Outlook

7. Notes

8. Tables and Figures

9. Tables and Figures

10. Tables and Figures

11. Tables and Figures

12. Tables and Figures

13. Tables and Figures

14. Tables and Figures

15. Tables and Figures

16. Tables and Figures

17. Tables and Figures

18. Tables and Figures