

Ford-Fulkerson Method

Flow maximization in a network (graph)
with capacities

Basic idea:

- Find a path from source to target that still has flow capacity (*augmenting path*)
- Add the maximum flow allowed along this path
- Repeat until *no further improvement*

Issues

1. How do we account for flow by
"cancellation" of existing
2. Does adding an augmenting path
lead to a legal flow?
What's a legal flow?
3. Will this process converge?
*Yes, capacities are integers or
rationals*
4. If so, will it lead to *maximum flow?*

Problem Formulation

Directed graph $G = (N, E)$

Two special nodes: s - source t - target

Assume for any node $v \in N$, there are paths

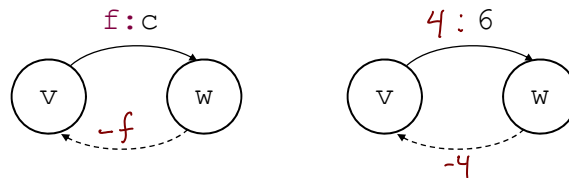
$s \rightsquigarrow v$ $v \rightsquigarrow t$ *unreachable nodes or "dead ends" - leave them out*

Capacity $c(v, w) \geq 0$

If (v, w) not an edge, then

$c(v, w) = 0$

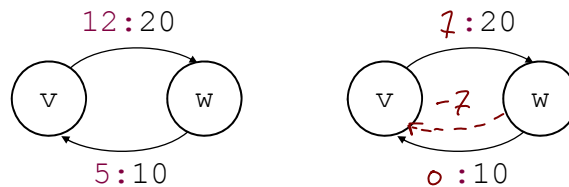
Flow $f(v, w)$ can be *positive or negative*



Detail

Assume no "useless" flows between nodes

Positive flow in only one direction



Legal Flow f

1. $f(v, w) \leq c(v, w)$ *capacity*
2. $f(v, w) = -f(w, v)$ *skew symmetry*
3. For any $v \neq s, t$

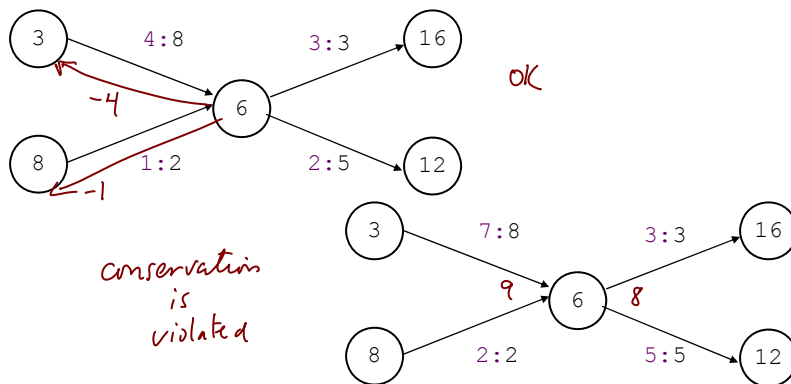
$$\sum_{w \in N} f(v, w) = 0$$
conservation

Inputs = Outputs

Consider node 6

$$f(6, 16) + f(6, 12) + f(6, 8) + f(6, 3)$$

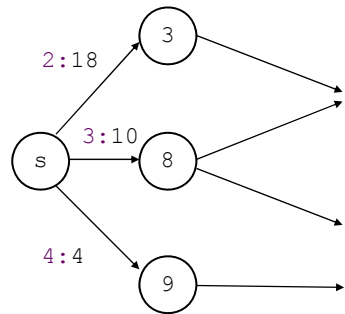
$$3 + 2 + (-4) + (-1) = 0$$



Value of Flow f

Total flow out of source

$$|f| = \sum_{w \in N} f(s, w)$$



$$2 + 3 + 4 = 9$$

Residual Capacity for a Flow

Residual Capacity between nodes v and w :

$$r \equiv C_f$$

$$r(u, v) = c(u, v) - f(u, v)$$



$$r(3, 7) = c(3, 7) - f(3, 7)$$

$$10 - 6 = 4$$

$$r(7, 3) = c(7, 3) - f(7, 3)$$

$$0 - (-6) = 6$$

Algorithm Design & Analysis

Residual Graph for Flow f

$R = (N, E')$
 $E' = \{ (v, w) \mid r(v, w) > 0 \}$

Capacities are residual capacities for f

Lecture Notes 5 David Maier 9

Algorithm Design & Analysis

Example

The top graph shows a network with nodes s, s1, s2, s3, s4, s5, s6, s7, s8, and t. Edges are labeled with flow:capacity. For example, s to s1 is 2:25, s1 to s2 is 0:9, s2 to s4 is 0:15, s4 to s6 is 2:4, s6 to t is 8:20, etc.

The bottom graph shows the same network after finding an augmenting path (s to s2 to s4 to s6 to t). The augmenting path is highlighted in green. Residual capacities are updated in red. For example, the edge s to s2 now has a residual capacity of 20, s2 to s4 has 15, s4 to s6 has 2, and s6 to t has 12. A red arrow points to the bottom graph with the text "residual graph for this flow".

Lecture Notes 5 David Maier 10

Algorithm Design & Analysis

Capacity of a Path

Minimum capacity edge here = 2
Add a flow of 2 along the path

12 → 14

Lecture Notes 5 David Maier **11**

Algorithm Design & Analysis

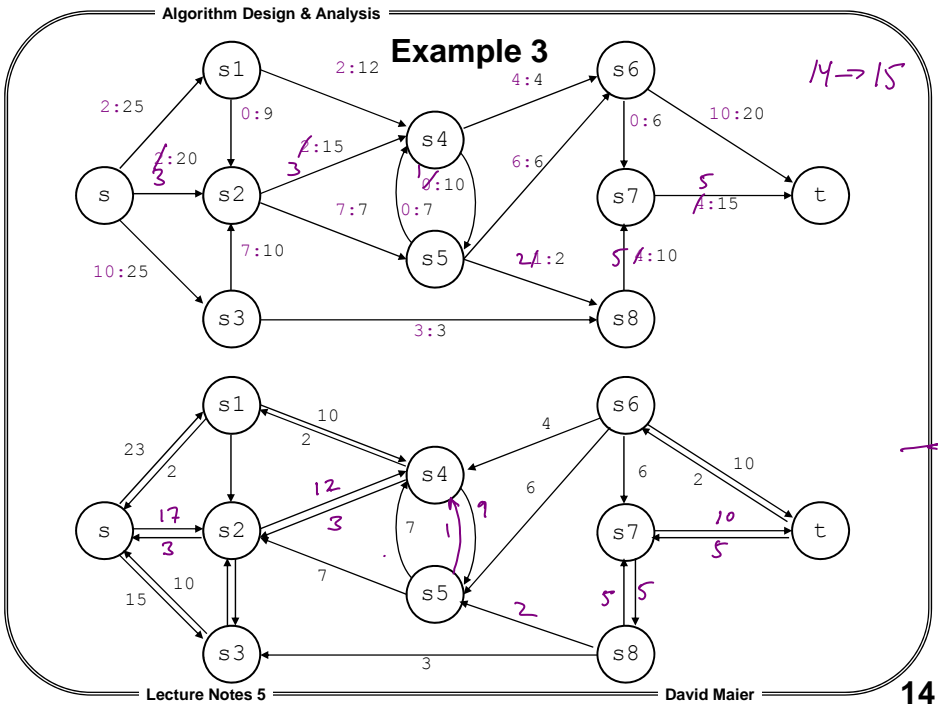
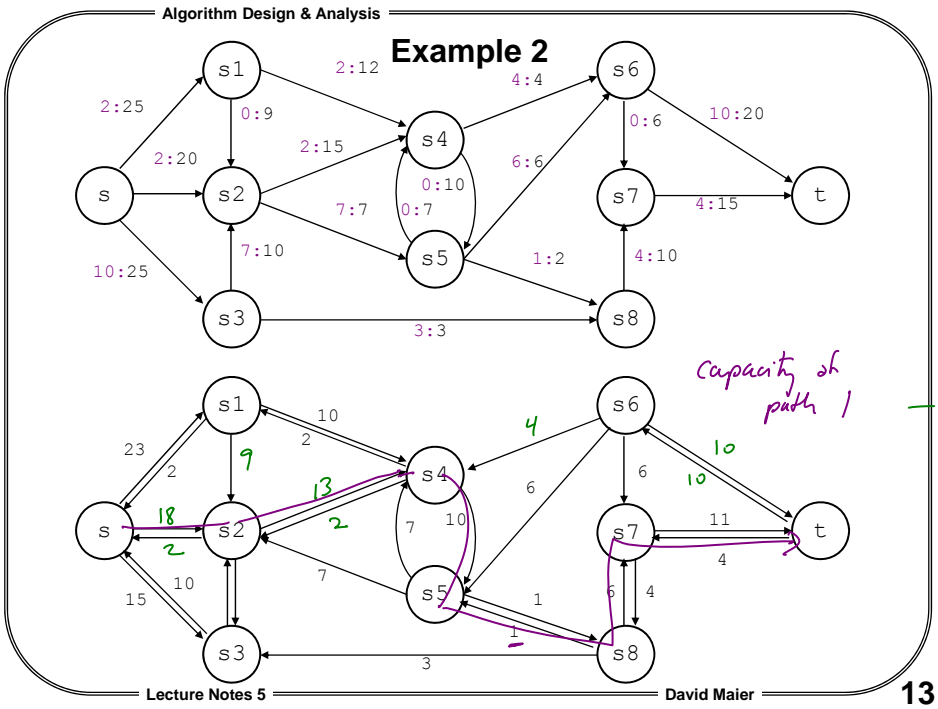
Is the Result a Legal Flow?

- Capacity? residual graph shows available capacity
- Skew symmetry? almost by definition
- Conservation? -4

If satisfied before, then satisfied after

Lecture Notes 5 David Maier **12**

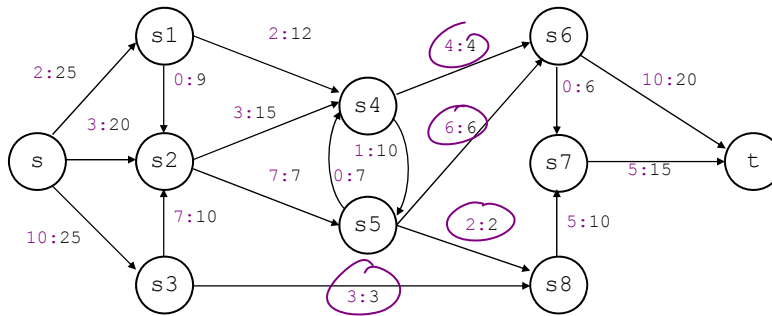
CS 410/584, Algorithm Design & Analysis, Lecture Notes 5



Is it a Maximum Flow?

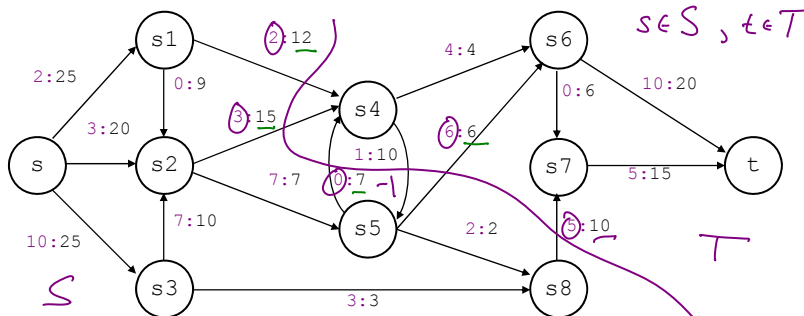
Would seem so:

Have a group of edges that divides s from t
and *that can't take more flow*



Cut of a Graph

Divide nodes of N into two groups S, T



Net flow across cut $\sum_{v \in S, w \in T} f(v, w)$

$2+3+0-1+6+5 = 15$

Capacity across cut $\sum_{v \in S, w \in T} c(v, w)$

$12+15+7+6+10 = 50$

Results

- Net flow across any cut is $|f|$
 - $|f|$ is bounded above by capacity of any cut
 - Max-flow/min-cut theorem
 1. f is maximum flow
 2. residual graph has no augmenting paths
 3. $|f|$ is capacity of some cut
- equivalent* ↻

2 ⇒ 3

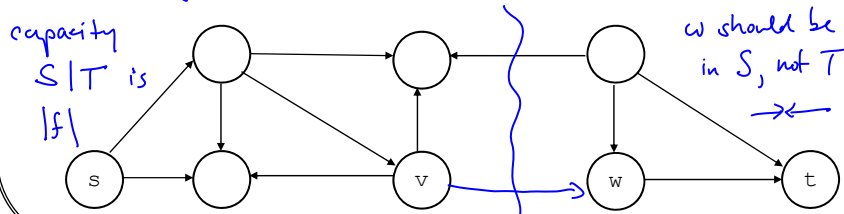
Consider residual graph R with no augmenting path

$$S = \{v \mid s \rightsquigarrow v \text{ in } R\} \quad T = N - S$$

Must have $s \in S \quad t \in T$

Claim that for (v, w) with $v \in S, w \notin T$, must have $r(v, w) = 0$ that is $f(v, w) = c(v, w)$

Suppose not. Then $r(v, w) > 0$. Then R has edge (v, w) . So w is reachable from s . So



Basic Implementation

Start with 0 flow

Repeat

Add flow along an augmenting path
as long as there is one

Does it always converge?

Yes, if capacities are integers. Flow grows by at least *one each iteration*

How long does it take?

If you pick augmenting paths arbitrarily

$O(|E| \cdot |f^*|)$

find an augmenting path (under $|E|$)
maximum flow (under $|f^*|$)

Edmonds-Karp Algorithm

Start with 0 flow

Repeat

Add flow along an augmenting path with
fewest edges BFS

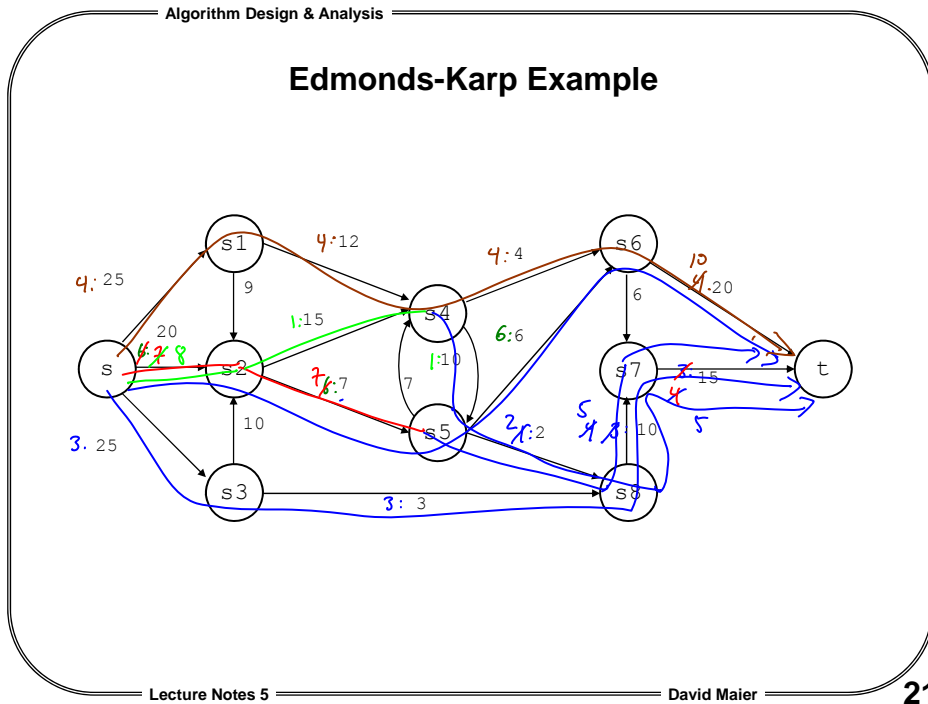
Trying to saturate shortest paths first

Time complexity no longer depends on value of maximum flow

$O(|N| \cdot |E|^2)$ time

Intuition: Length of shortest path to a node v in residual graph *never decreases*

- Each addition of flow increases distance to one node
- Distance to a node v can be increased at most *$|E|$*



Algorithm Design & Analysis

Matrix Multiplication

For a divide-and-conquer approach, we want an $n \times n$ matrix to look like a matrix of smaller matrices

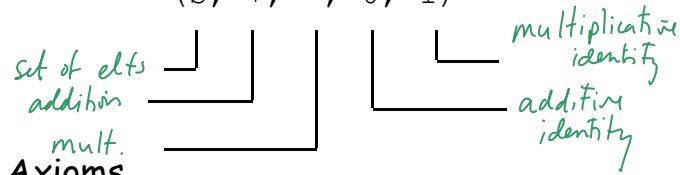
$$\begin{pmatrix} 1 & 3 & 21 & 9 \\ 4 & 12 & 36 & 35 \\ 42 & 6 & 10 & 53 \\ 2 & 32 & 12 & 60 \end{pmatrix} \begin{pmatrix} 41 & 2 & 6 & 14 \\ 3 & 11 & 7 & 18 \\ 40 & 13 & 16 & 29 \\ 7 & 19 & 63 & 72 \end{pmatrix}$$

How is a matrix of matrices like a matrix of integers?

Lecture Notes 5 David Maier **22**

Ring

(S, +, ·, 0, 1)



Axioms

1. +, · associate $(a+b)+c = a+(b+c)$
2. + commutes $a+b = b+a$
3. · distributes over +
 $(a+b) \cdot c = a \cdot c + b \cdot c$ $a \cdot (b+c) = a \cdot b + a \cdot c$
4. 0 is identity for + $a+0 = 0+a = a$
5. 1 is identity for · $a \cdot 1 = 1 \cdot a = a$
6. every a in S has an additive inverse $-a$
 $(a+(-a)) = 0 = ((-a)+a)$

Example Ring

Integers modulo n, \mathbb{Z}_n

- [a] is $[n-a]$

0 is $[0]$

1 is $[1]$

\mathbb{Z}_4 $[0], [1], [2], [3]$

- [1] is $[4-1] = [3]$

[1] + [3] = [4] = [0]

Algorithm Design & Analysis

Key Example

for example
elts are integers

$(M_n, +_n, \cdot_n, 0_n, 1_n)$

$n \times n$ matrices
over a ring R

matrix addition

matrix mult.

Identiy matrix

matrix of all zeros

is a ring.

Inverse of $A = (a_{ij})$ is $-A = (-a_{ij})$

Get weird

$R_{2, n/2}$ is

all 2×2 matrices whose
entries are $\frac{n}{2} \times \frac{n}{2}$ matrices

$\left(\begin{array}{cc} \left(\begin{array}{cc} 1^r & 1^r \end{array} \right) & \left(\begin{array}{cc} 1^r & 1^r \end{array} \right) \\ \left(\begin{array}{cc} 1^r & 1^r \end{array} \right) & \left(\begin{array}{cc} 1^r & 1^r \end{array} \right) \end{array} \right)$

Lecture Notes 5 David Maier **25**

Algorithm Design & Analysis

Going Between M_n and $R_{2, n/2}$

A, B, C in M_n where *where n is even*

A', B', C' in $R_{2, n/2}$

$A' \left(\begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \right) \left(\begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22} \end{array} \right) = B'$

$\left(\begin{array}{cc} A_{11} \cdot B_{11} + A_{12} \cdot B_{21} & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ A_{21} \cdot B_{11} + A_{22} \cdot B_{21} & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{array} \right) = \left(\begin{array}{cc} C_{11} & C_{12} \\ C_{21} & C_{22} \end{array} \right) = C'$

reduce $n \times n$ matrix mult to $n/2 \times n/2$
matrix ops

Lecture Notes 5 David Maier **26**

Complexity

Let $M(n)$ be the number of scalar operations ($+$, \cdot) to multiply two $n \times n$ matrices over \mathbb{R} *on elts over ring R*

Divide-and-conquer doesn't buy much

$M(1) = 1$ $M(n) = 8 \cdot M(\frac{n}{2}) + 4 \cdot (\frac{n}{2})^2$ $8M(\frac{n}{2}) + n^2$
add

In general, if

$M(n) = m M(\frac{n}{2}) + an^2$

then $M(n) \leq kn^{\log_2 m}$ for some k ,
 provided $m > 4$

Master Theorem § 4.3

Complexity 2

If $m = 8$, as before

$M(n) = O(n^{\lg 8}) = O(n^3)$

If $m = 7$

$M(n) = O(n^{\lg 7}) \approx O(n^{2.81})$

Need 2×2 multiplication with 7 scalar multiplications

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

e.g., $c_{22} = a_{21}b_{12} + a_{22}b_{22}$

Strassen's Algorithm

$$\begin{aligned}
 m_1 &= (a_{12} - a_{22})(b_{21} + b_{22}) \\
 \bullet m_2 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\
 \bullet m_3 &= (a_{11} - a_{21})(b_{11} + b_{12}) \\
 m_4 &= (a_{11} + a_{12})b_{22} \\
 \bullet m_5 &= a_{11}(b_{12} - b_{22}) \\
 m_6 &= a_{22}(b_{21} - b_{11}) \\
 \bullet m_7 &= (a_{21} + a_{22})b_{11}
 \end{aligned}$$

Don't want to use commutativity of multiplication

More Algebra

$$\begin{aligned}
 C_{11} &= m_1 + m_2 - m_4 + m_6 & M(n) &= 7 \cdot M\left(\frac{n}{2}\right) + 18n^2 \\
 C_{12} &= m_4 + m_5 \\
 C_{21} &= m_6 + m_7 \\
 C_{22} &= m_2 - m_3 + m_5 - m_7
 \end{aligned}$$

$$\begin{aligned}
 C_{22} &= (a_{11} + a_{22})(b_{11} + b_{22}) && a_{11}b_{11} + a_{22}b_{11} + a_{11}b_{22} + a_{22}b_{22} \\
 &- (a_{11} - a_{21})(b_{11} + b_{12}) && + (a_{11}b_{11} + a_{11}b_{12} + a_{21}b_{11} + a_{21}b_{12}) \\
 &+ a_{11}(b_{12} - b_{22}) && + a_{11}b_{12} - a_{11}b_{22} \\
 &- (a_{21} + a_{22})b_{11} && + (a_{21}b_{11} + a_{22}b_{11}) \\
 &&& a_{22}b_{22} + a_{21}b_{12} = a_{21}b_{12} + a_{22}b_{22}
 \end{aligned}$$

Another Use of Matrix of Matrices

Connect matrix inversion with matrix multiplication

Inverse of matrix A , denoted A^{-1} , is the unique matrix such that $AA^{-1} = I$

I is the identity matrix

Inverse doesn't always exist - if it exists, it's unique

Example

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} -3 & 2 \\ 2 & -1 \end{pmatrix} = \begin{pmatrix} -3+4 & 2-2 \\ -6+6 & 4-3 \end{pmatrix}$$

Inversion vs. Multiplication

Assuming matrix inversion behaves reasonably, it has the same order as multiplication

Let $I(n) = \# \text{ scalar ops to invert an } n \times n \text{ matrix}$

- assume
1. $I(n) \in \Omega(n^2)$
 2. $I(3n) \in O(I(n))$

Time to multiply two $n \times n$ matrices, $M(n)$, is in $O(I(n))$.

Reduce Matrix Multiply to an Inversion Problem

Suppose we have A, B to multiply,
where both are $n \times n$

Construct a $3n \times 3n$ matrix D

$$\begin{pmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{pmatrix}$$

$n \times n$ Identity.

Claim that D^{-1} is

$$\begin{pmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{pmatrix}$$

Check

$DD^{-1} =$

$$\begin{pmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{pmatrix} \begin{pmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{pmatrix}$$

$$\begin{pmatrix} I_n \cdot I_n & I_n \cdot A + A \cdot I_n & I_n \cdot AB + A \cdot (-B) \\ 0 & I_n \cdot I_n & I_n \cdot (-B) + B \cdot I_n \\ 0 & 0 & I_n \cdot I_n \end{pmatrix} = \begin{pmatrix} I_n & 0 & 0 \\ 0 & I_n & 0 \\ 0 & 0 & I_n \end{pmatrix}$$

I_{3n}